
opennaru
Apache Camel 소개
(Enterprise Integration Pattern)
KHAN
apm

opennaru

패턴 언어

" 각각의 패턴은 우리를 **둘러싸고 있는 환경**에서 반복적으로 나타나는 특정한 **문제**와 그에 대한 **해결책**을 설명한다.
그리고 그 해결책은 계속 사용될 수 있기 때문에
동일한 과정을 반복할 필요가 없다."

- 크리스토퍼 알렉산더 'A Pattern Language'

opennaru

패턴은

- 자주 접하는 '**문제**'와 그 '**해법**'을 쌍으로 구성해,
  특정 문제에 대한 **검증된 스키마**를 제공하여 손쉽게 **재활용**
- 설계나 구현에 대한 **Communication**시 공통 **Vocabulary** 역할
- **경험**을 통해 축적된 실무 지식을 효과적으로 요약하고 **전달**
- 추상적인 **원칙**과 **코드** 작성에 가이드 역할

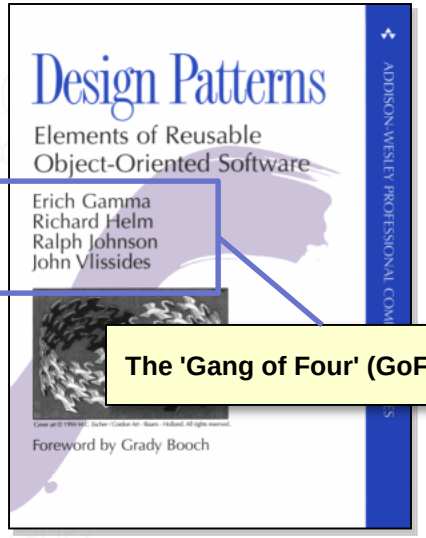
opennaru

Application Performance Management

Enterprise Integration Pattern

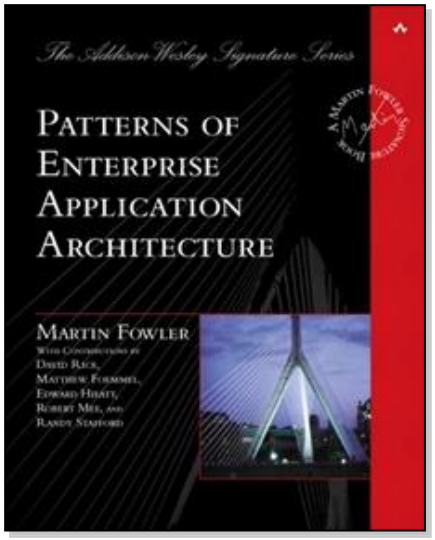
KHAN
[apm]
[gbw]

Design Patterns

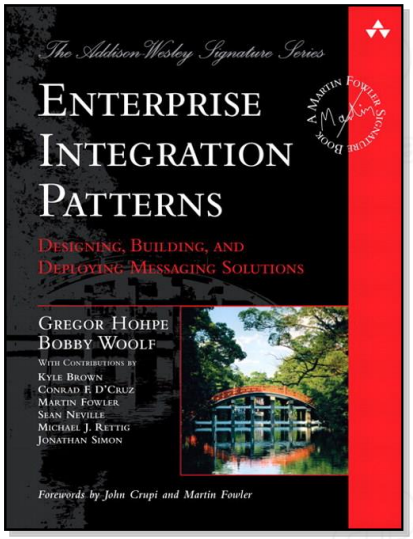


The 'Gang of Four' (GoF) book

- Adapter
- Facade
- Proxy
- Observer
- ...



- Gateway
- Model/View/Controller
- Registry
- Repository
- Service Layer
- ...

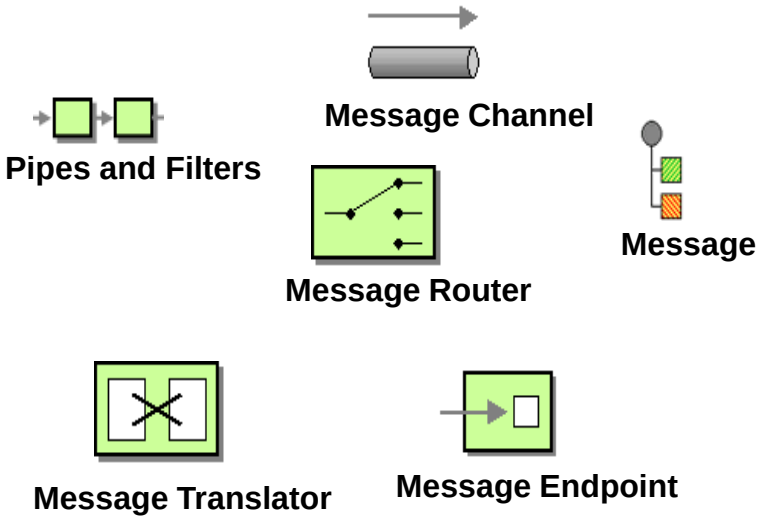


- Aggregator
- Content Filter
- Recipient List
- Request/Reply
- Publish-Subscribe Channel
- ...

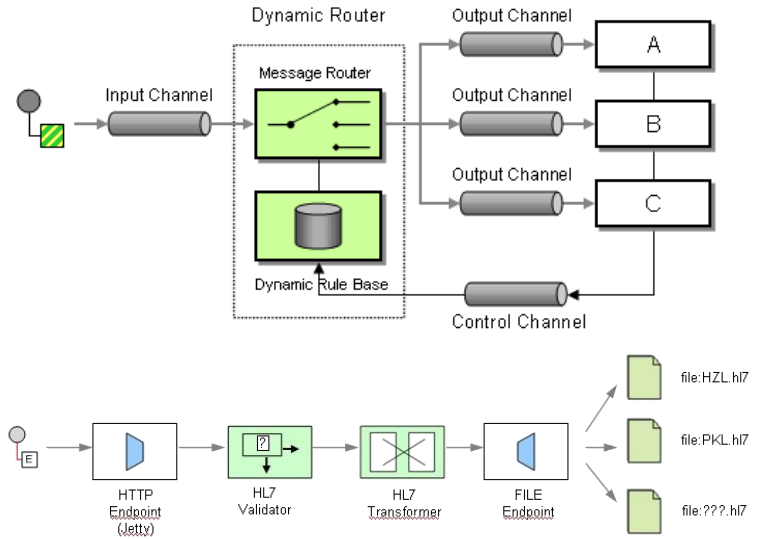
- Common architectural patterns that have emerged over time
- Technology-agnostic – Technologies can change (C/C++/Java/Scala?) over time, but patterns remain consistent
- Documenting them and naming them provides a common vocabulary for design discussions
- Architecture of entire enterprise can often be described in terms of combinations of these pat terns

- 범용적인 설계 패턴을 GoF 디자인 패턴으로서 정의한 것처럼, 엔터프라이즈 통합의 패턴을 정의

필요한 기능을 정의

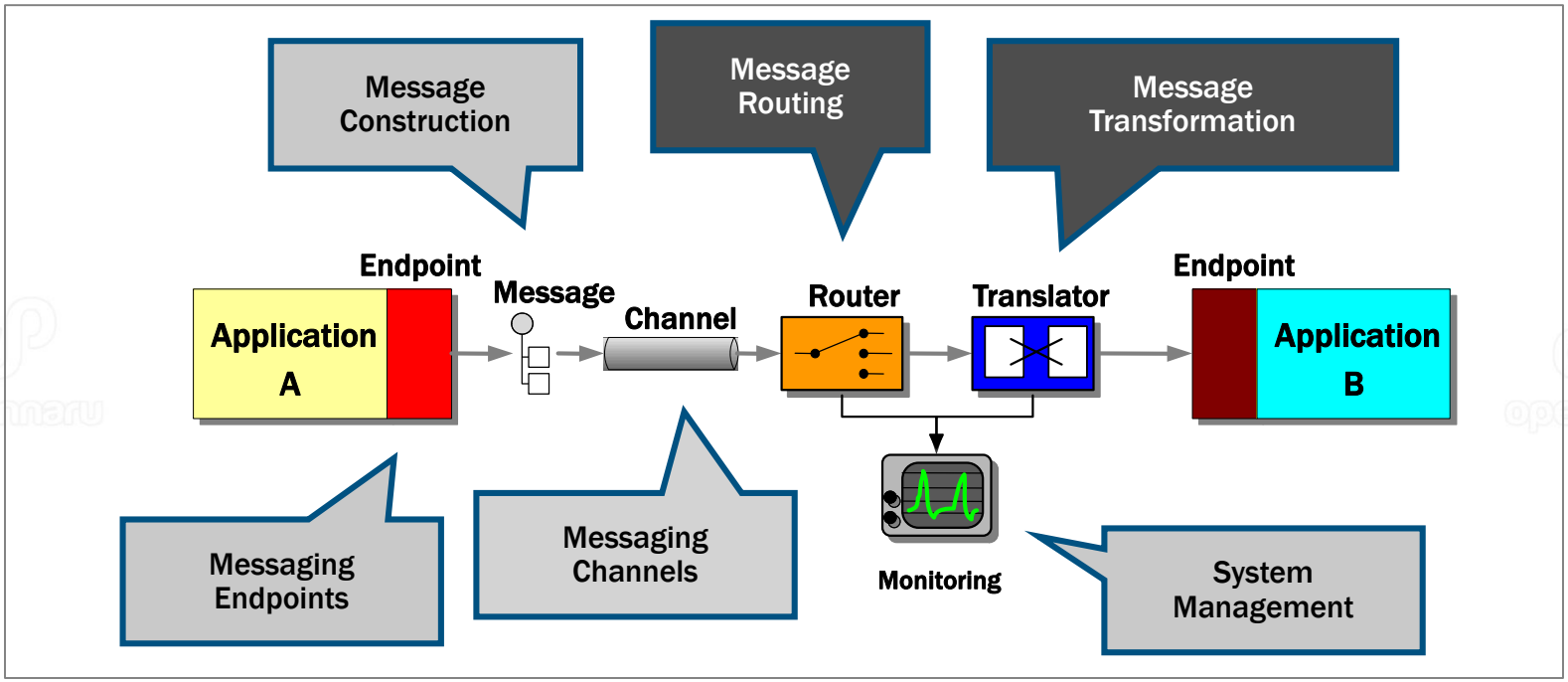


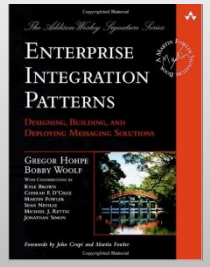
이용 시 주의 사항을 정리



Enterprise Integration Patterns

- Visual pattern language for message-based enterprise integration solutions
- Pattern language comprises of 65 patterns in 6 categories

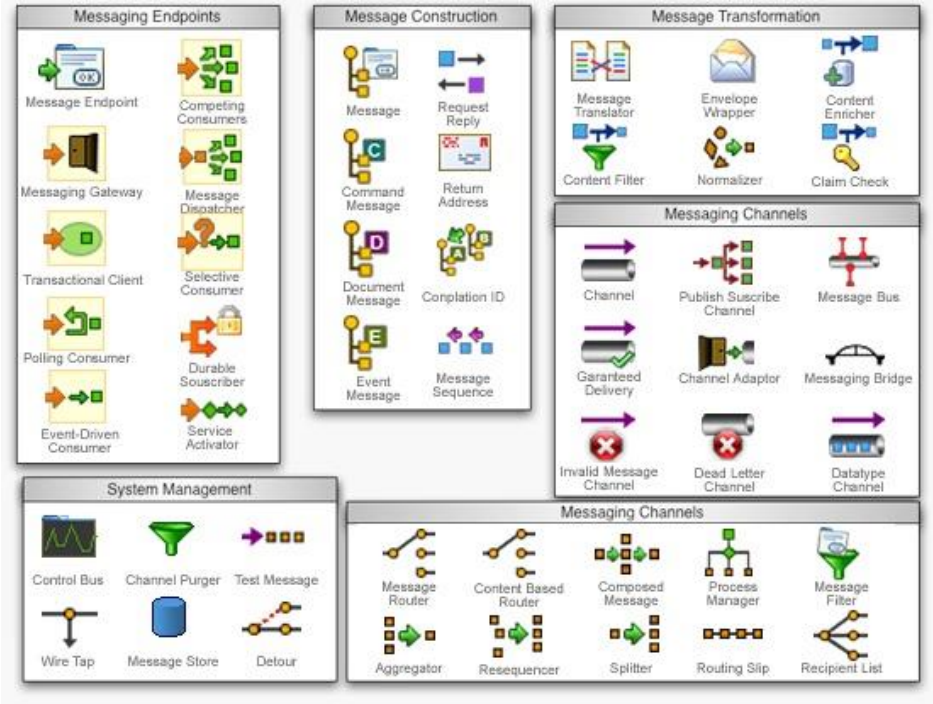




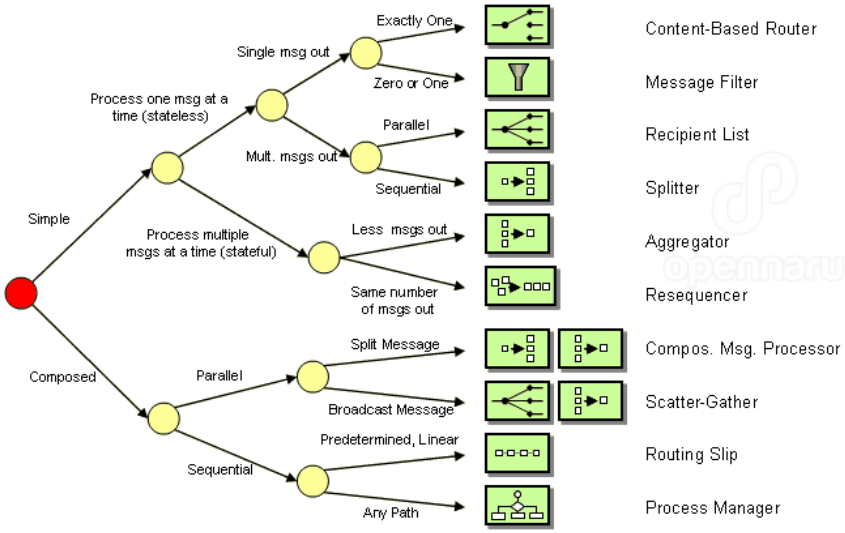
- A Book by Gregor Hohpe and Booby Woolf (2003)
- Patterns and Recipes for common integration problems
- Message Centric
- Used as the basis for all the major integration products
- Should be the the first thing to reference when starting an integration project

What about SOA or ESB or Web Service Patterns?

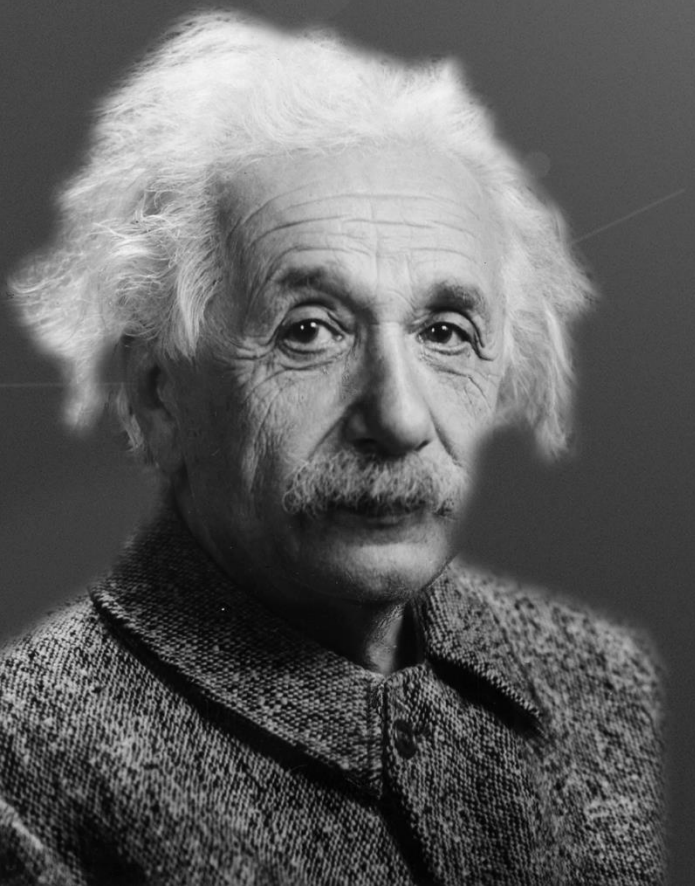
- 기업 분산 컴퓨팅 환경에서 시스템 간 통신, 데이터 교환 아키텍처의 디자인(밑그림) 문제에 대한 해결책을 문서화하는 정식 방법



Pattern Catalog



opennaru



"'The measure of intelligence is the ability to change"

- Albert Einstein

opennaru

Application Performance Management

Camel Introduction

KHAN
[apm]
[gbw]

Why the name Camel?

Concise
Application
Messaging
Exchange
Language

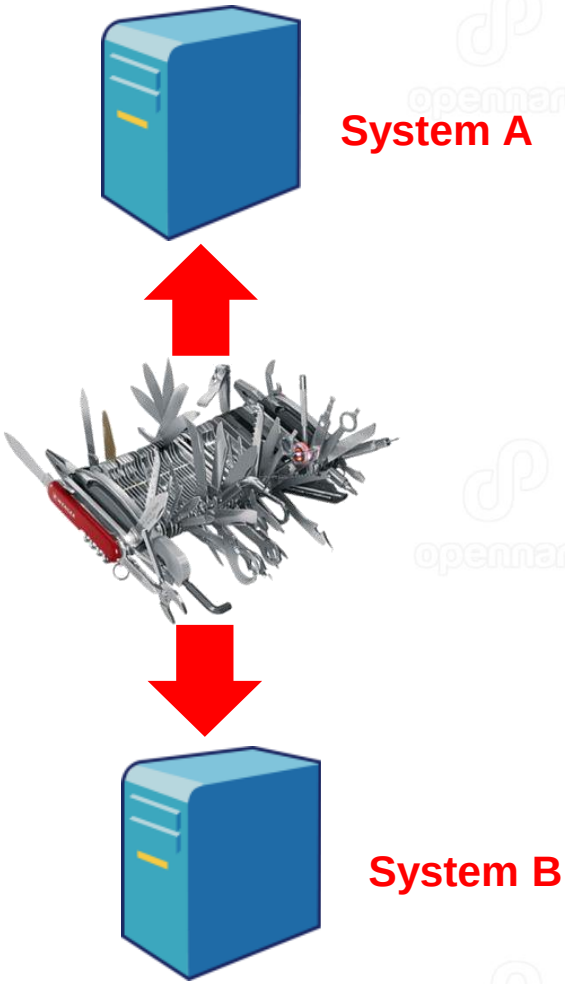
Apache Camel is a powerful Open Source Integration Framework based on known Enterprise Integration Patterns

http://camel.apache.org/why-the-name-camel.html

What is Apache Camel

Summary

- Integration framework
- Enterprise Integration Patterns (EIP)
  - DSL을 이용한 Integration 표현
- 손쉬운 구성 (endpoints as URIs)
  - 메시지 채널, Endpoint
- No heavy specification
- Light Weight - Any 컨테이너
  - 거의 모든 자바 기반 컨테이너에 배포
- 100+ Camel 컴포넌트, 타입 변환



The Swiss army knife of Open source integration

Apache Camel 이란?

- **Apache Camel는 EIP를 구현하는 통합 프레임워크**
  - **어플리케이션 설계와 개발을 용이해짐**
- **EIP를 엔드포인트와 프로세서로 구현**
  - **엔드 포인트**
    - 외부 연계 인터페이스(File나 JMS 등)
    - Camel 엔드 포인트는 URI로 표현

    예) file:data/input
- **프로세서**
  - **데이터 변환이나 라우팅 그리고 메시지 처리를 하는 것**
  - **프로세서는 여러 개를 연결하여 처리**
- **메시지 플로우를 라우트(route)로 정의**
  - **라우트는 엔드 포인트와 프로세서의 연쇄에 의해 구성되는 메시지 플로우**
  - **Apache Camel에서는 라우트는 DSL (Domain Specific Language)로 표현**
    - Java 또는 XML

Code 로 Camel 이해하기 – File 복사

순수 자바 코드로 작성

```
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStream;

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 {
```

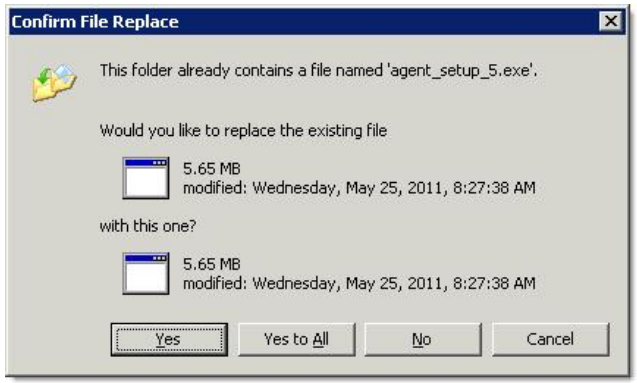
Java DSL 로 작성

```
import org.apache.camel.CamelContext;
import org.apache.camel.builder.RouteBuilder;
import org.apache.camel.impl.DefaultCamelContext;


public class FileCopierWithCamel {

    public static void main(String args[])
                                    throws Exception {
        // create CamelContext
        CamelContext context
                    = new DefaultCamelContext();

        // add our route to the CamelContext
        context.addRoutes(new RouteBuilder() {
            public void configure() {
                from("file:data/inbox?noop=true")
                .to("file:data/outbox");
            }
        });
        // start the route and let it do its work
        context.start();
        Thread.sleep(10000);
        // stop the CamelContext
        context.stop();
    }
}
```

Code 로 Camel 이해하기 – 복잡한 File 복사 요건들

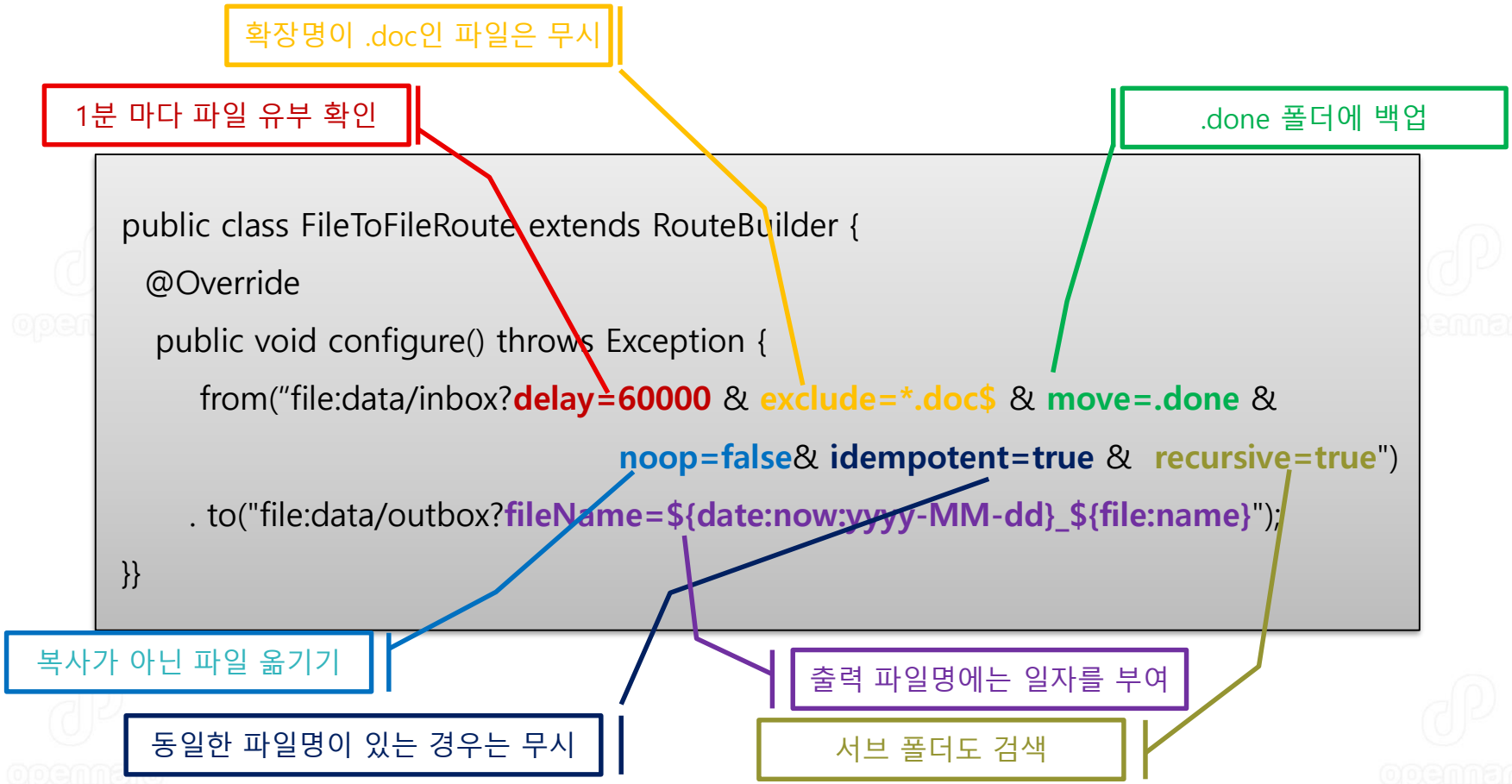
그런데 , 보통은 좀 더 요건이 복잡...

- ☐ 1분 마다 파일이 있는지 체크하는 것
- ☐ 날짜로 파일명 만들기
- ☐ .doc 로 확장자가 붙은 파일은 무시하기
- ☐ 동일한 파일명이 있는 경우는 무시하는 것
- ☐ 서브 폴더도 검색
- ☐ 복사가 아니고 이동으로 변경하고 백업도 수행
- ☐ 등등 ....



Apache Camel - File Component

- **The File component provides access to file systems, allowing files to be processed by any other Camel Components or messages from other components to be saved to disk.**

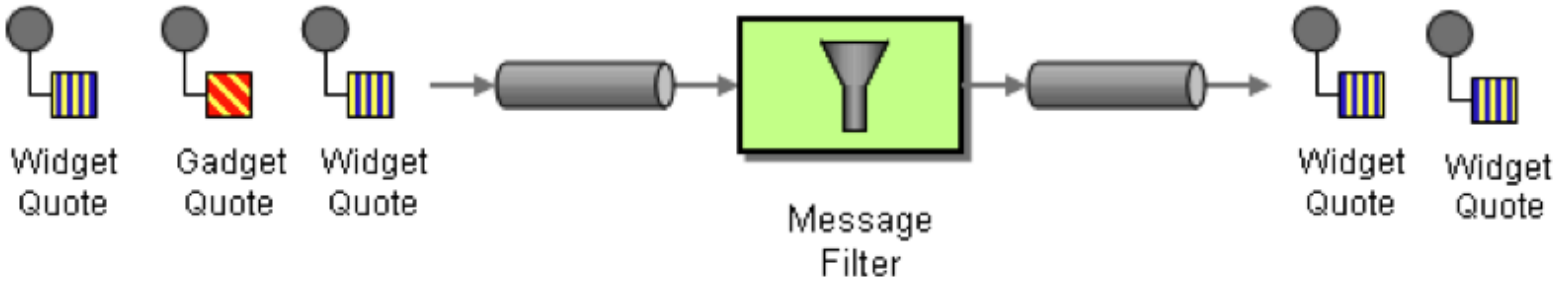


Contents Base Routing

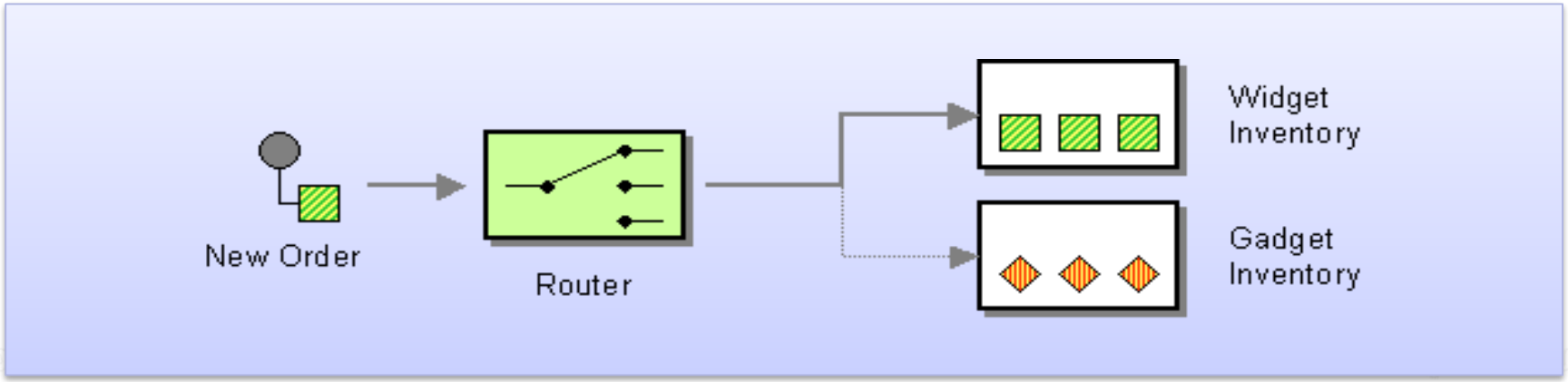
Apache
ActiveMQ

WebSphereMQ

from(A) **filter(message)** **to(B)**



Content Based Router

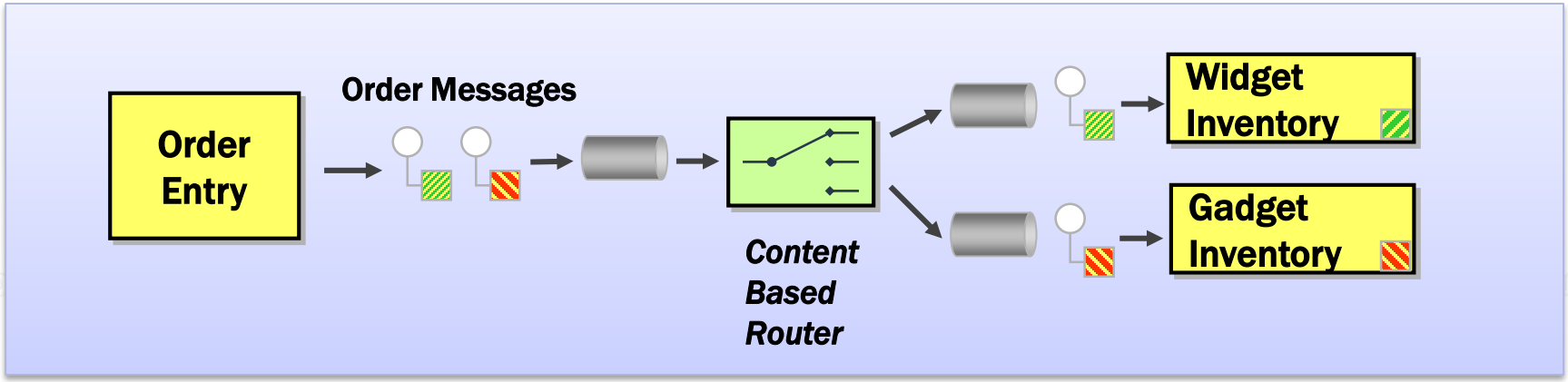



```
Endpoint A = endpoint("activemq:queue:quote");
Endpoint B = endpoint("mq:quote");
Predicate isWidget = xpath("/quote/product = 'widget'");

from(A).filter(isWidget).to(B);
```

Camel Example

```
import org.apache.camel.builder.RouteBuilder;

public class FilterRoute extends RouteBuilder {

  public void configure() throws Exception {
    Endpoint A = endpoint("activemq:queue:quote");
    Endpoint B = endpoint("mq:quote");
    Predicate isWidget = xpath("/quote/product = 'widget'");

    from(A).filter(isWidget).to(B);
  }
}
```

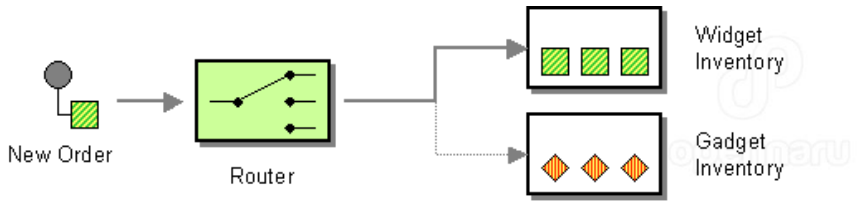
- Camel provides an embedded DSL (in Java & Spring) for implementing enterprise integration patterns
  - The DSL uses URIs to define endpoints which are combined by form integration flows




```
from("activemq:topic:Quotes")
  .filter().xpath("/quote/product = 'widget'")
    .to("mqseries:WidgetQuotes")
  .filter().xpath("/quote/product = 'gadget'")
          .to("mqseries:GadgetQuotes");
```

DSL (Domain Specific Language)을 이용한 Integration 표현(1)

Enterprise Integration Pattern
- Content Based Router



Content Based Router – XML DSL

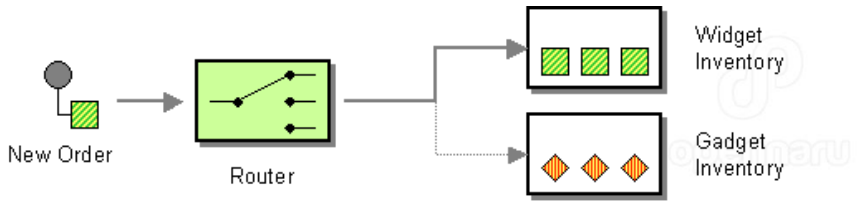
```
<camelContext>
  <route>
    <from uri="activemq:NewOrders"/>
      <choice>
        <when>
          <xpath>/order/product = 'widget'</xpath>
            <to uri="activemq:Orders.Widgets"/>
        </when>
        <otherwise>
          <to uri="activemq:Orders.Gadgets"/>
        </otherwise>
    </choice>
  </route>
</camelContext>
```

Content Based Router – Java DSL

```
from("activemq:NewOrders")
.choice()
.when().xpath("/order/product = 'widget'")
.to("activemq:Orders.Widget")
.otherwise()
.to("activemq:Orders.Gadget");
```

DSL (Domain Specific Language)을 이용한 Integration 표현(2)

Enterprise Integration Pattern
- Content Based Router



Endpoints - File

Use file instead

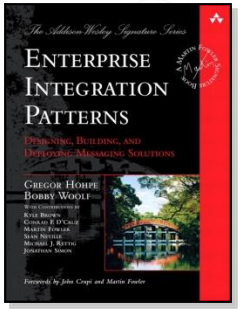
```
from("file:inbox/orders")
.choice()
.when().xpath("/order/product = 'widget'")
.to("activemq:Orders.Widget")
.otherwise()
.to("activemq:Orders.Gadget");
```

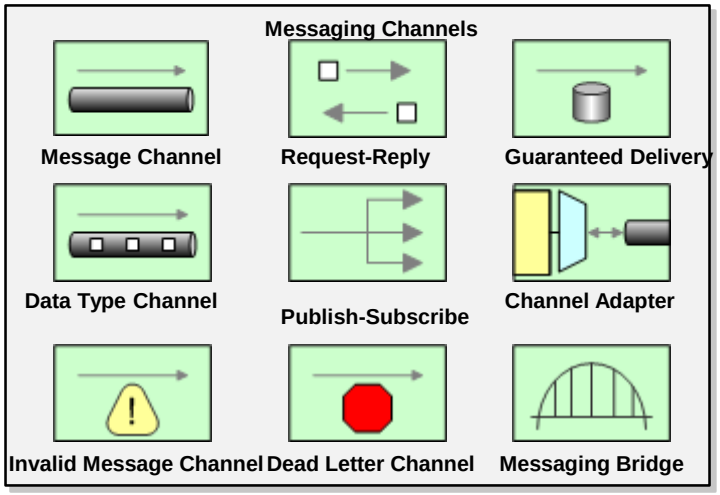
Endpoints - parameters

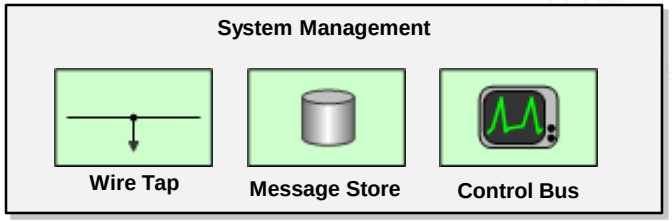
parameters

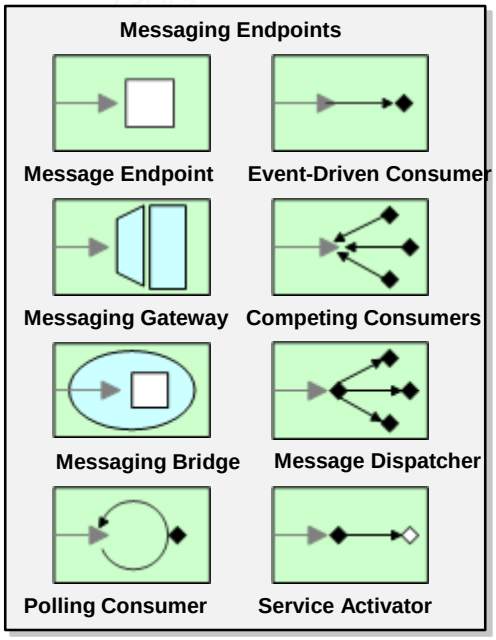
```
from("file:inbox/orders?delete=true")
.choice()
.when().xpath("/order/product = 'widget'")
.to("activemq:Orders.Widget")
.otherwise()
.to("activemq:Orders.Gadget");
```

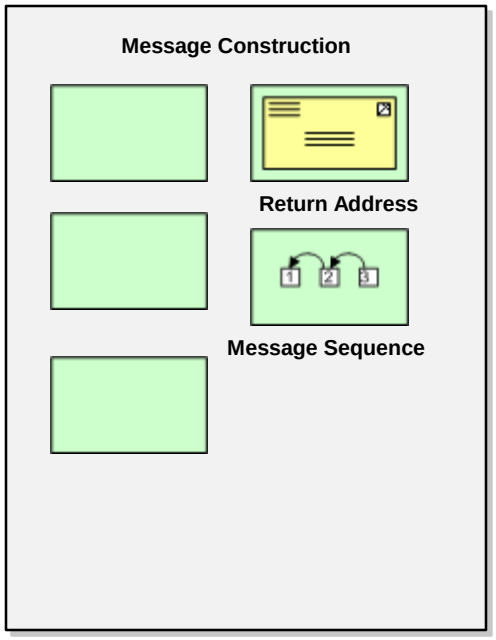
Camel Patterns

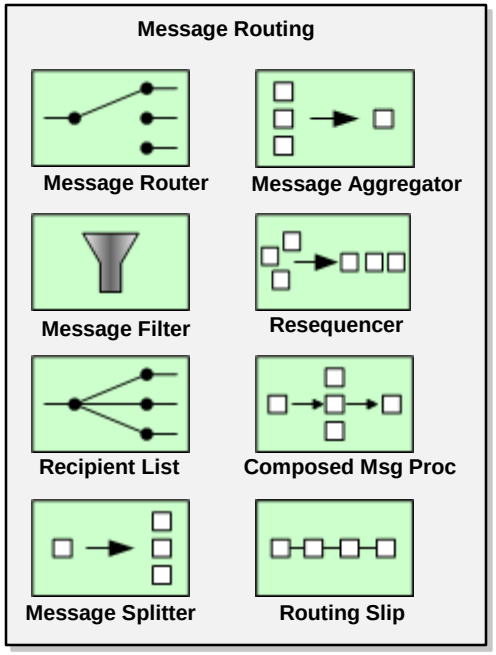


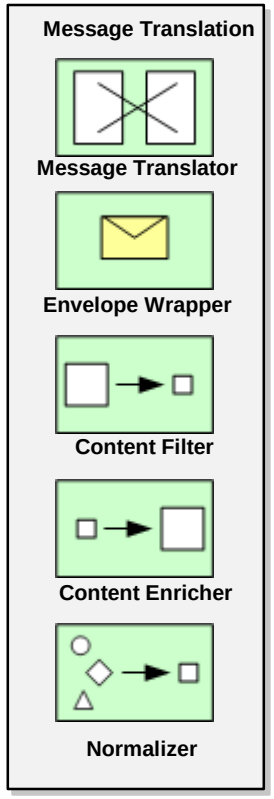












http://camel.apache.org/enterprise-integration-patterns.html

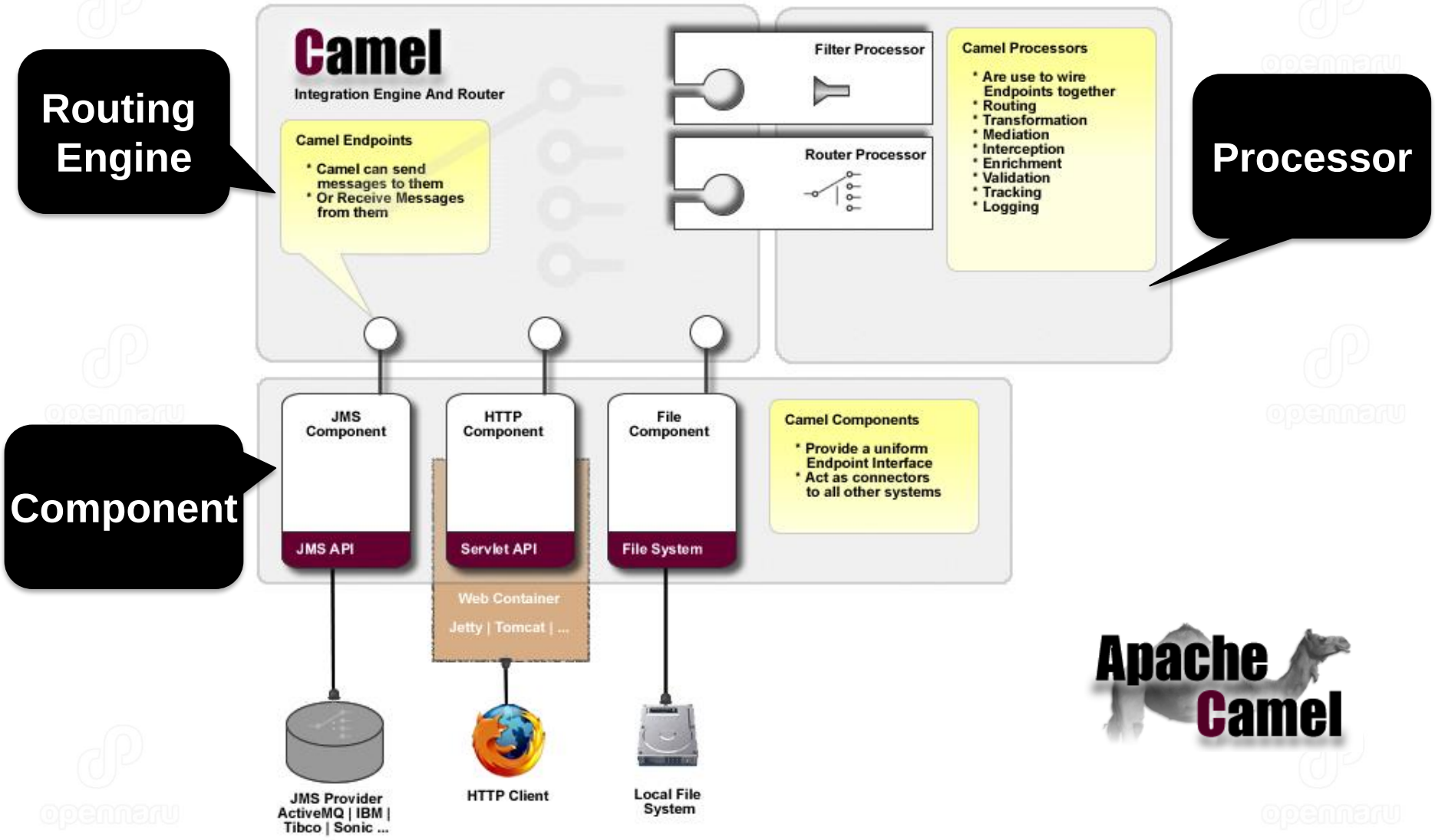
opennaru

Application Performance Management

Camel Architecture

KHAN
apm
gbw

Apache Camel - Architecture



Camel Component – Custom Component

- **Over 120+ Components**
- **Camel includes the following Component implementations via URIs.**

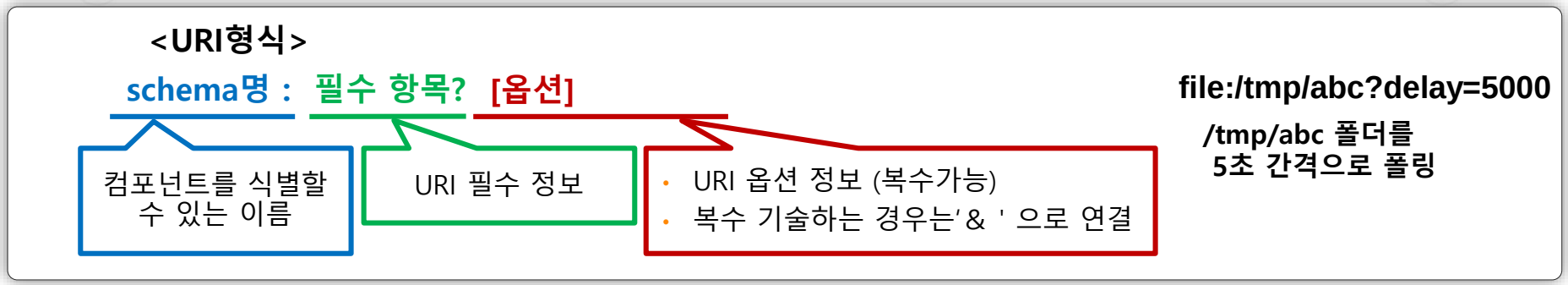
activemq	cxf	flatpack	jasypt
activemq-journal	cxfrs	freemarker	javaspace
amqp	dataset	ftp/ftps/sftp	jbi
atom	db4o	gae	jcr
bean	direct	hdfs	jdbc
bean validation	ejb	hibernate	jetty
browse	esper	hl7	jms
cache	event	http	jmx
cometd	exec	ibatis	jpa
crypto	file	irc	jt/400

- SSH component / camel-ssh — `ssh:[username[:password]@]host[:port][?options]`
- Twitter / camel-twitter — `twitter://[endpoint]?[options]`
- iBATIS / camel-ibatis — `ibatis://statementName`

- **http://camel.apache.org/components.html**

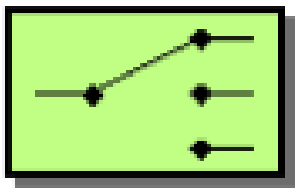
Camel Component – URI Format

- 컴포넌트는 URI로 지정

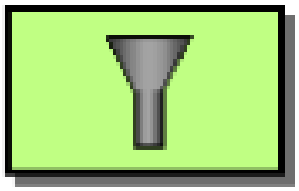


컴포넌트	프로토콜	URI
CXF	웹서비스	cxf:address[serviceClass=...]
File	File	file:fileOrDirectoryName[options]
FTP/SFTP/FTPS	FTP/SFTP/FTPS	ftp://[username@]host[:port]/directoryname[options]
JDBC	JDBC	jdbc:dataSourceName[options]
Jetty	HTTP 서버	jetty:http://hostname[:port][/resourceUri][options]
JMS	JMS	jms:[queue:|topic:]destinationName[options]
AWS-S3	AmazonSimple StorageService(S3)	aws-s3://bucketname[options]
GHttp	URLfetchservice (GoogleAppEngine)	ghttp://hostname[:port][/path][options] ghttp:///path[options]
Log	-	log:loggingCategory[options]

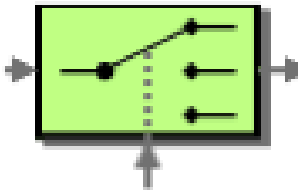
CameI 에서 제공하는 패턴



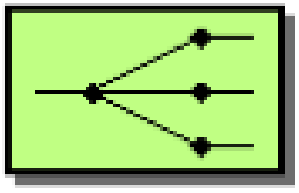
Content Based Router



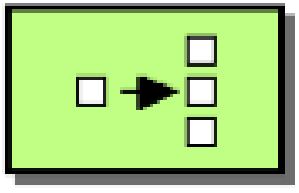
Message Filter



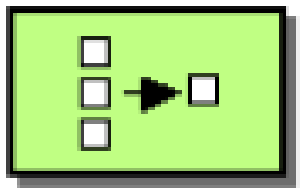
Dynamic Router



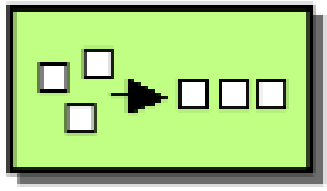
Recipient List



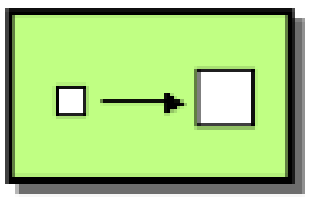
Splitter



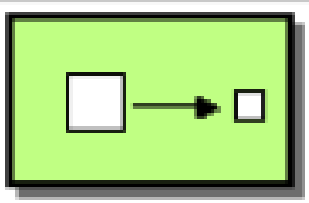
Aggregator



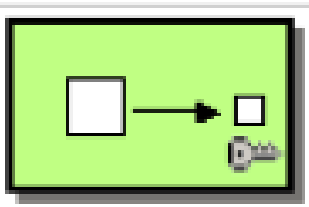
Resequencer



Content Enricher



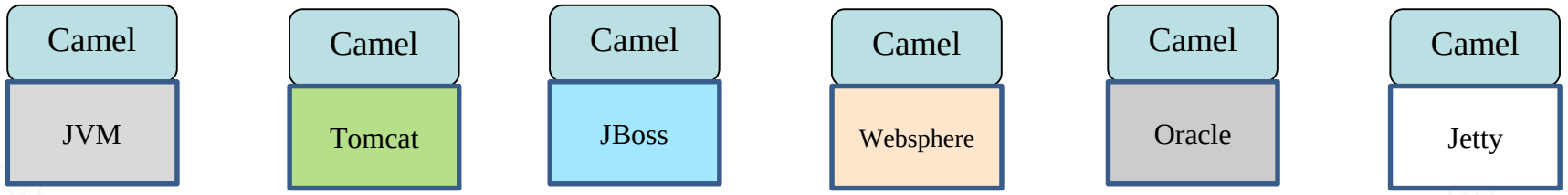
Content Filter



Claim Check

Deploy it wherever you need

- Replaceable container
- Developer friendly DSL
- Productivity
- Built-in scalability primitives



Cloud

Spring Container

Application Server

Standalone

OSGi

Web Container

opennaru

Application Performance Management

KHAN
[apm]
[gbw]

제품이나 서비스에 관한 문의

콜 센터 : 02-469-5426 (휴대폰 : 010-2243-3394)

전자 메일 : sales@opennaru.com