2014/15

ETL mit EAI

4BHIT
ADLER/KARIC

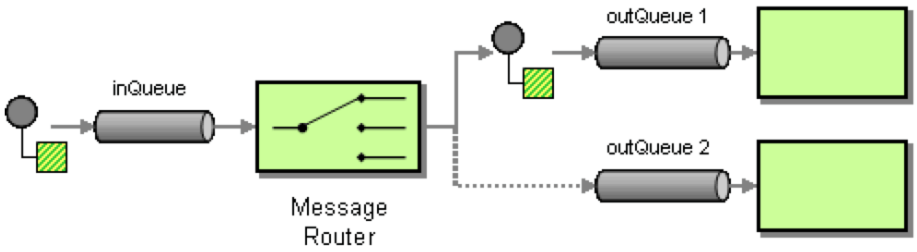
TGM WIEN XX | 22.02.2015

Inhaltsverzeichnis

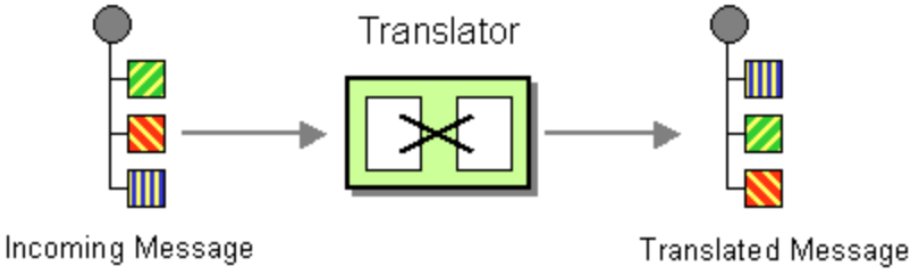
1. Identifikation und Beschreibung der EAI Patterns

Routing

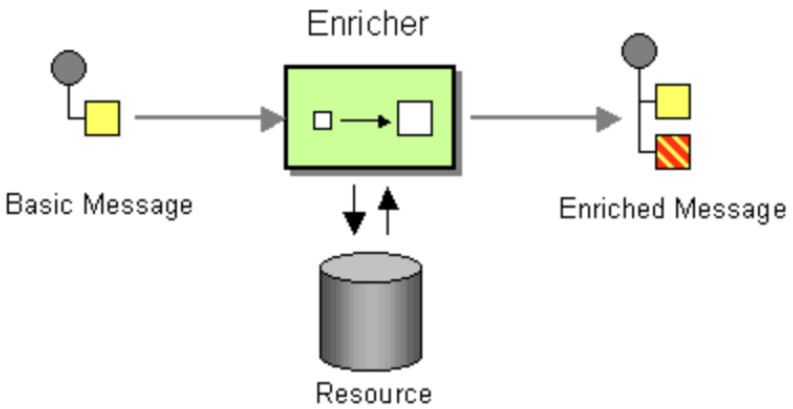
- Message Router
  Legt einen speziellen Filter, einen Message Router, welcher von einem Message Channel die Nachricht holt und es an einen anderen Message Channel Kanal in Abhängigkeit von Bedingungen weiterleitet.



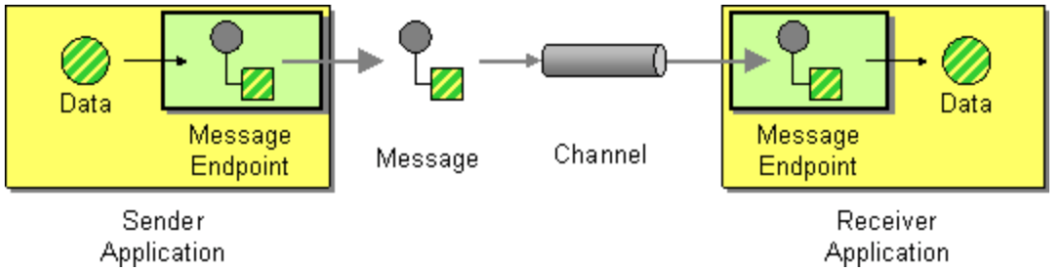
- **Transformation**
  - Message Translator
    Verwenden einen speziellen Filter, Message Translator, zwischen anderen Filtern oder Anwendungen und wandeln das Format in ein anderes.



- **Enrichment**
  - Content Enricher
    Verwendet einen speziellen Transformer, Content Enricher, um eine externe Datenquelle mit fehlenden Informationen zu ergänzen.



- **Message**
  - Message Endpoint
    Verbindet eine Application mit einem Messaging Channel mittels Message Endpoint. Ein Client des Messaging-System, kann die Anwendung verwenden, um Nachrichten zu senden oder zu empfangen.



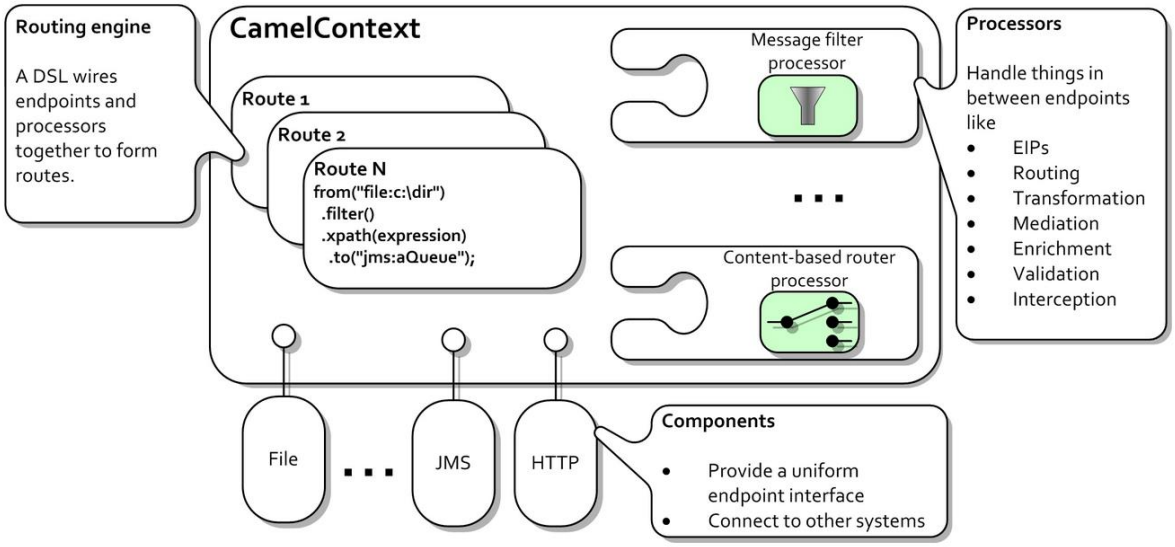


Figure 1: High level view of Camel's architecture (from Camel in Action).

Processors are used to manipulate and mediate messages in between Endpoints. All of the EIPs are defined as Processors or sets of Processors. As of writing, Camel supports over 40 patterns from the EIP book and many other useful Processors. (Anstey)

Quellen:

(Stan), (Hohpe)

2. Funktionsweise von Apache Camel

Apache Camel basiert auf Enterprise Integration Patterns - Entwurfsmuster welche für den Entwurf von Enterprise Application Integration und Message Oriented Middleware basierten Systemen geschaffen wurden. Apache Camels Bean Binding unterstützt dabei Plain Old Java Objects und JavaBeans. Dadurch integriert es einfach mit Dependency Injection Frameworks wie Spring oder Google Guice.

Apache Camel verwendet Uniform Resource Identifiers und kann somit direkt mit unterschiedlichen Transport- und Messageprotokollen wie beispielsweise HTTP, JMS oder AMQP umgehen. Es kann so beispielsweise mit JBI, SCA, Apache ActiveMQ, RabbitMQ, Apache MINA oder Apache CXF zusammenarbeiten. Somit kann basierend auf der Apache Camel Programmierschnittstelle gearbeitet werden, obwohl die darüber angesprochenen Komponenten technologisch unterschiedliche Schnittstellen verwenden.

Apache Camel wird häufig zusammen mit Apache ServiceMix (Enterprise Service Bus), Apache CXF (Web Service Framework) und Apache ActiveMQ (Java Message Service Provider) in SOA Infrastruktur Projekten eingesetzt. Darüber hinaus wird oft auch Apache MINA (Framework für Netzwerkapplikationen) zusammen mit Apache Camel verwendet.

(SunGard CSA LLC),

(J2EE)

3. Inbetriebnahme des Beispiels

Der erste Schritt ist das Downloaden vom Apache Camel von der offiziellen Website *http://camel.apache.org/download.html*

Hier wählt man die neueste Version für Unix/Linux –Systeme aus (in unserem Fall 2.14.1). Dieses Verzeichnis entpackt man nun im Zielordner.

Unix/Linux/Cygwin Distribution (2.14.x branch)	apache-camel-2.14.1.tar.gz

```
schueler@debian:~/bin/EAI/apache-camel-2.14.1$ ls
doc  examples  lib  LICENSE.txt  NOTICE.txt  README.txt
schueler@debian:~/bin/EAI/apache-camel-2.14.1$
```

Im Verzeichnis findet man mehrere txt-Dateien sowie drei Unterordner. Wir wollen das etl-example zum Laufen bringen daher ist der example-Ordner für uns von Bedeutung. Hier navigieren wir zum etl-example (siehe unten)

```
schueler@debian:~/bin/EAI/apache-camel-2.14.1/examples/camel-example-etl$ ls
derby.log  pom.xml  README.txt  src  target
schueler@debian:~/bin/EAI/apache-camel-2.14.1/examples/camel-example-etl$
```

Wie wir im README lesen können müssen wir um das Beispiel zu starten zwei Befehle ausführen:

mvn compile

mvn camel:run

Damit man diese Befehle ausführen kann ist es aber nötig maven zu installieren (maven2 hat für uns nicht ganz funktioniert!). Wenn dies gemacht wurde können wird den ersten Befehl *mvn compile* ausführen. Wir sehen nun folgende Ausgabe:

```
schueler@debian:~/bin/EAI/apache-camel-2.14.1/examples/camel-example-etl$ mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building Camel :: Example :: ETL 2.14.1
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] --- maven-bundle-plugin:2.3.7:cleanVersions (versions) @ camel-example-etl ---
[INFO]
[INFO] --- maven-remote-resources-plugin:1.5:process (default) @ camel-example-etl ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ camel-example-etl ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 7 resources
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default) @ camel-example-etl ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 7 resources
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-compiler-plugin:2.5.1:compile (default-compile) @ camel-example-etl ---
[INFO] Nothing to compile - all classes are up to date
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 7.521s
[INFO] Finished at: Sun Feb 22 11:49:57 CET 2015
[INFO] Final Memory: 12M/40M
[INFO] ------------------------------------------------------------------------
schueler@debian:~/bin/EAI/apache-camel-2.14.1/examples/camel-example-etl$
```

Wir erhalten ein BUILD SUCCESS und führen nun den Befehl *mvn camel:run* aus:

Wir sehen zunächst (nach mehreren Infos) die Ausgabe einer Person (als XML-Struktur). Man sieht auch von wo diese Daten kommen: row2.xml und wie sie intern repräsentiert werden nämlich als PersonDocument.

```
h=154, CamelFileAbsolute=false, CamelFileParent=src/data, CamelFileNameConsumed=row2.xml, CamelFileName=row2.
xml, CamelFileNameOnly=row2.xml}, BodyType:org.apache.camel.example.etl.PersonDocument, Body:<?xml version="1
.0" encoding="UTF-8" standalone="yes"?>
<person user="hiram">
    <firstName>Hiram</firstName>
    <lastName>Chirino</lastName>
    <city>Tampa</city>
</person>
```

Danach folgt ein SELECT aus der Datenbank und anschließend findet der CustomerTransformer eine passende CustomerEntity (hiram). Zuletzt wird noch das Customer Objekt erzeugt. Derselbe Prozess passiert auch bei der 2.Person (James)

```
5780  camel  TRACE  [Camel (camel) thread #0 - file://src/data] openjpa.jdbc.SQL - <t 1375051624, conn 142111
9201> executing prepstmnt 849517840
SELECT t0.id, t0.city, t0.firstName, t0.phone, t0.street, t0.surname,
       t0.userName, t0.zip
    FROM customer t0
    WHERE (t0.userName = ?)
[params=?]
5782  camel  TRACE  [Camel (camel) thread #0 - file://src/data] openjpa.jdbc.SQL - <t 1375051624, conn 142111
9201> [0 ms] spent
2015-02-22 11:54:11,702 [file://src/data] INFO  CustomerTransformer           - Found a matching CustomerEnt
ity Customer[userName: hiram firstName: Hiram surname: Chirino] having the userName hiram.
2015-02-22 11:54:11,703 [file://src/data] INFO  CustomerTransformer           - Created object customer: Cus
tomer[userName: hiram firstName: Hiram surname: Chirino]
```

Hier sieht man dann die Ausgabe des neu erzeugten Customers hiram (id=1)

```
02445165-0-5}, BodyType:org.apache.camel.example.etl.CustomerEntity, Body:<?xml version="1.0" encoding="UTF-8
" standalone="yes"?>
<customer id="1">
    <userName>hiram</userName>
    <firstName>Hiram</firstName>
    <surname>Chirino</surname>
    <city>Tampa</city>
</customer>
```

Hier die Ausgabe des 2. erzeugten Customers James (id=2)

```
mpl@3f0842de}, BodyType:org.apache.camel.example.etl.CustomerEntity, Body:<?xml version="1.0" encoding="UTF-8
" standalone="yes"?>
<customer id="2">
    <userName>james</userName>
    <firstName>James</firstName>
    <surname>Strachan</surname>
    <city>London</city>
</customer>
```

Die beiden Personen (PersonDocument) wurden also durch einen CustomerTransformer in Customer gewandelt (CustomerEntity).

4. Literaturverzeichnis

Anstey, J. (n.d.). *Open Source Integration with Apache Camel and How Fuse IDE Can Help.* Retrieved 02 22, 2015, from http://java.dzone.com: http://java.dzone.com/articles/open-source-integration-apache

Hohpe, G. (n.d.). *Enterprise Integrations Patterns.* Retrieved 02 22, 2015, from http://www.eaipatterns.com: http://www.eaipatterns.com/docs/jaoo_hohpeg_enterpriseintegrationpatterns.pdf

J2EE, J. a. (n.d.). *Apache Camel with Sql Example .* Retrieved 02 22, 2015, from https://www.youtube.com: https://www.youtube.com/watch?v=z6TizMFTIvI

Stan, E. (n.d.). *Enterprise Integration Patterns with Apache Camel.* Retrieved 02 22, 2015, from http://de.slideshare.net: http://de.slideshare.net/ieugen222/eip-cu-apache-camel

SunGard CSA LLC. (n.d.). *How Apache Camel works.* Retrieved 02 22, 2015, from http://help.eclipse.org: http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.stardust.docs.camel%2Fhtml%2Fcamel%2Fcamel-introduction.html