



---

# ETL WITH EAI

---

SYT / DEZSYS09



22. FEBRUAR 2015

ERCEG, KRITZL  
4AHITT

## Inhalt

1. Aufgabenstellung.....	2
2. Requirementanalyse mit Aufwandabschätzung.....	3
3. anschließende Endzeitaufteilung .....	4
3.1 Erceg .....	4
3.2 Kritzl.....	4
3.3 Gesamtsumme .....	4
4. Arbeitsdurchführung .....	5
4.1 Identifikation und Beschreibung der EAI Patterns .....	5
4.1.1 File Transfer .....	5
4.1.2 Shared Database.....	5
4.1.3 Pipes and Filters.....	6
4.1.4 Message Translator .....	6
4.1.5 Message Endpoint .....	7
4.1.6 Document Message.....	7
4.1.7 Polling Consumer .....	8
4.2 Beschreibung der Technologien .....	9
4.2.1 Definition von ETL.....	9
4.2.2 Definition von EAI .....	9
4.2.3 Beschreibung und Funktionsweise von Apache Camel .....	10
4.3 Inbetriebnahme des Beispiels .....	11
5. Testbericht.....	13
6. Lessons learned .....	14
7. Quellenangaben .....	14
8. Abbildungsverzeichnis.....	15

*Github-Link:* <https://github.com/mkritzl-tqm/DezSys09-EAI>

*Github-Tag:* `erceg_kritzl_dezsys09_v1`

## 1. Aufgabenstellung

### Gruppenaufgabe (2 Leute)

*"The ETL (Extract, Transform, Load) is a mechanism for loading data into systems or databases using some kind of Data Format from a variety of sources; often files then using Pipes and Filters, Message Translator and possible other Enterprise Integration Patterns. So you could query data from various Camel Components such as File, HTTP or JPA, perform multiple patterns such as Splitter or Message Translator then send the messages to some other Component.*

*To show how this all fits together, try the ETL Example."* [1]

ETL ist ein wichtiger Prozess bei einem Datawarehouse. Zeigen Sie wie Enterprise Integration Patterns [2] dabei eingesetzt werden können (8 Punkte, nur jene, die in dem Beispiel vorkommen). Verwenden Sie dazu das ETL Example [3]. Dokumentieren Sie die Implementierung sowie alle notwendigen Schritte ausführlich in einem Protokoll (8 Punkte). Fügen Sie den verwendeten Code nach den Metaregeln an und geben Sie alles als ZIP-Archiv (Gesamtes Framework mit Anleitung, wie das System gestartet werden kann) ab.

### Resources

[1] Extract Transform Load (ETL); Apache Camel;

Online: <http://camel.apache.org/etl.html>; abgerufen 13.02.2015

[2] Enterprise Integration Patterns; G.Hohpe, B.Woolf; 2003;

Online: <http://www.enterpriseintegrationpatterns.com/toc.html>; abgerufen 13.02.2015

[3] Extract Transform Load (ETL) Example; Apache Camel;

Online: <http://camel.apache.org/etl-example.html>; abgerufen 13.02.2015

## 2. Requirementanalyse mit Aufwandabschätzung

Arbeitspaket	Zuständige Person	Geschätzte Zeit	Erledigt
Example zum Laufen bringen	Erceg, Kritzl	60 min	x
Funktionsweise von Apache Camel beschreiben	Erceg, Kritzl	50 min	x
Code dokumentieren	Erceg, Kritzl	100 min	x
Herausfinden, welche Enterprise Integration Patterns im Example verwendet wurden	Erceg, Kritzl	200 min	x
Unit-Tests	Erceg, Kritzl	200 min	x
<i>Gesamt</i>		<b>610 Minuten (10 h 10 min)</b>	

### 3. anschließende Endzeitaufteilung

#### 3.1 Erceg

Arbeit	Datum	Zeit in Minuten
Example zum Laufen bringen	13.02.2015	10 Minuten
Example zum Laufen bringen	18.02.2015	15 Minuten
Example zum Laufen bringen	19.02.2015	90 Minuten
Code dokumentieren	19.02.2015	180 Minuten
Verwendete Patterns beschrieben	20.02.2015	120 Minuten
Verwendete Patterns beschrieben	22.02.2015	90 Minuten
Beschreibung der Technologien	22.02.2015	50 Minuten
Testung des Programms	01.03.2015	120 Minuten
<i>Gesamt</i>	<i>01.03.2015</i>	<b>675 Minuten (11 h 15 min)</b>

#### 3.2 Kritzl

Arbeit	Datum	Zeit in Minuten
Example zum Laufen bringen	13.02.2015	10 Minuten
Example zum Laufen bringen	18.02.2015	15 Minuten
Example zum Laufen bringen	19.02.2015	90 Minuten
Code dokumentieren	19.02.2015	180 Minuten
Verwendete Patterns beschrieben	20.02.2015	120 Minuten
Verwendete Patterns beschrieben	22.02.2015	90 Minuten
Beschreibung der Technologien	22.02.2015	50 Minuten
Maven Testreport, Code-Coverage	01.03.2015	120 Minuten
<i>Gesamt</i>	<i>01.03.2015</i>	<b>675 Minuten (11 h 15 min)</b>

#### 3.3 Gesamtsumme

Insgesamt haben wir für diese Übung **22 Stunden und 30 Minuten** benötigt. Geschätzt wurden 10 Stunden und 10 Minuten, daher lag unsere Einschätzung ziemlich daneben.

## 4. Arbeitsdurchführung

### 4.1 Identifikation und Beschreibung der EAI Patterns

#### 4.1.1 File Transfer

Damit mehrere Applikationen miteinander agieren können und die jeweiligen Informationen austauschen können, wird als Pattern das File Transfer angeboten. Dabei stellen eine oder mehrere Dateien den zentralen Mittelpunkt des Patterns dar, die in einem bestimmten Intervall erneuert werden. „Integrators“ sind für die korrekte Transferierung der Dateien zu verschiedenen Anforderungen zuständig. [1]

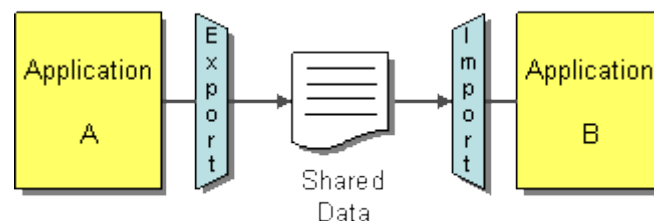


Abbildung 1: File Transfer [Abb1]

Im ETL-Example kommt das Pattern bei der Transferierung der XML-Files sowohl am Anfang als auch am Ende zu Stande. Die XML-Files der Personen könnten von einem vorgeschalteten System generiert worden sein und die erstellten XML-Files der Kunden von einem anderen System erneut verwendet werden.

#### 4.1.2 Shared Database

Eine zentrale Datenbank gewährt mehreren verschiedenen Applikationen Zugriff auf die Daten, die aufgrund der zentralen Speicherung sehr einfach zu verwalten sind. [2]

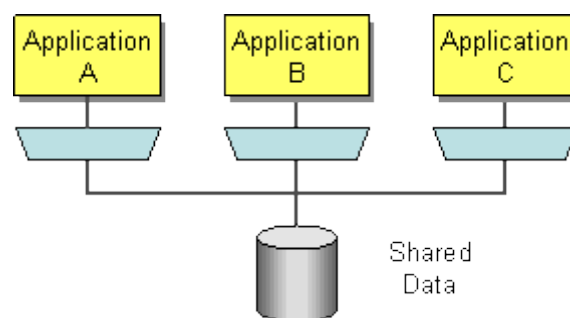


Abbildung 2: Shared Database [Abb2]

Im ETL-Example kommt das Pattern bei der Speicherung der Kundendaten in die Datenbank zur Anwendung. Dabei greift sowohl die Seite, bei der die Personendaten aus dem bereits vorhandenen XML-Files herausgelesen und in der Datenbank abgelegt werden, als auch die Seite, bei der die Kundendaten aus der Datenbank gelesen und wiederum abgespeichert werden.

#### 4.1.3 Pipes and Filters

Dieses Pattern ermöglicht die Zerstückelung eines größeren Tasks in mehrere kleinere Teile, die sequentiell abgearbeitet werden. Diese Teile werden Filter genannt und sind durch eine Pipe miteinander verbunden. Ein bestimmter Filter bekommt eine Message, diese wird von ihm verarbeitet und danach zu dem nächsten Filter weitergeleitet. [3]

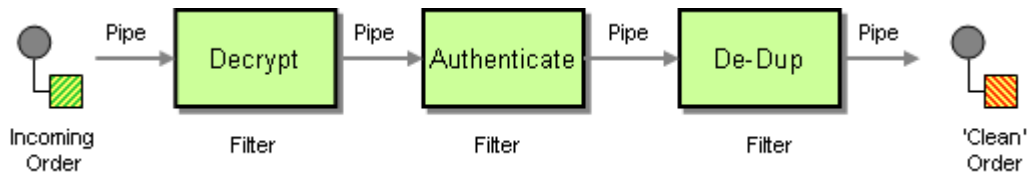


Abbildung 3: Pipes and Filters [Abb3]

Bei dem Example wird ein XML-File einer Person zuerst gelesen, dann in ein POJO transformiert und schlussendlich mittels JPA in die Datenbank geschrieben wird.

#### 4.1.4 Message Translator

Bei diesem Pattern wird eine ankommende Nachricht vom Translator mittels eines Filters in eine andere Form gebracht. Dies kommt zu Stande, wenn der Sender und der Empfänger nicht die gleichen Interfaces für die Kommunikation verwenden und dadurch die gesendete Nachricht vom Empfänger nicht verstanden werden würde. [4]

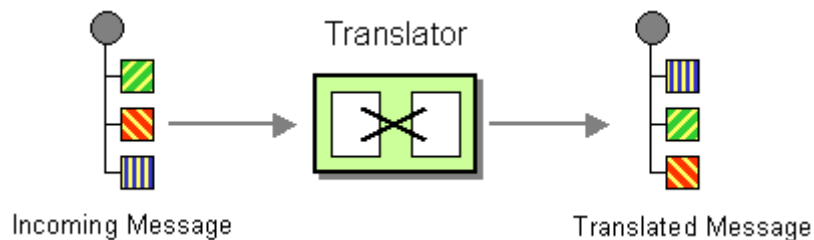


Abbildung 4: Message Translator [Abb4]

Der Customer Translator stellt in dem Example den Message Translator dar und übersetzt die Informationen der Person zu Informationen des Kunden.

#### 4.1.5 Message Endpoint

Der Message Endpoint ist sowohl für das Versenden, als auch das Empfangen von Nachrichten zuständig. Dabei muss die wirkliche Applikation nicht viel von dem Format der Nachricht oder gar der Verbindung zu der anderen Applikation wissen. Somit kümmert sich der Message Endpoint lediglich um die Kommunikation und richtige Formatierung der Nachricht, damit diese korrekt bei der anderen Applikation ankommt. [5]

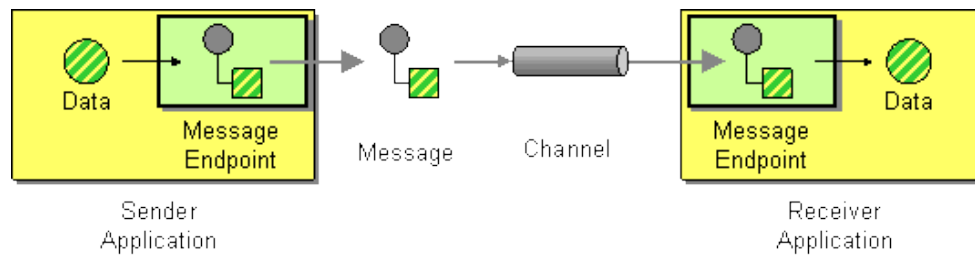


Abbildung 5: Message Endpoint [Abb5]

Die Klasse `EtlRoutes` dient bei dem Example der richtigen Weiterleitung der ermittelten Daten. So wird zum Beispiel definiert, wo die XML-Files der Personen liegen, wie diese transformiert (Person → Kunde) und abgelegt (Datenbank) werden sollen.

#### 4.1.6 Document Message

Ein bestimmtes Dokument steht dem Empfänger hierbei zur Verfügung, wobei der Empfänger selber entscheiden kann, wie dieses Dokument weiter verwendet wird. Dabei stellt das Dokument nichts weiter als Daten dar, die in keinem Bezug mit anderen Applikationen mehr stehen. [6]

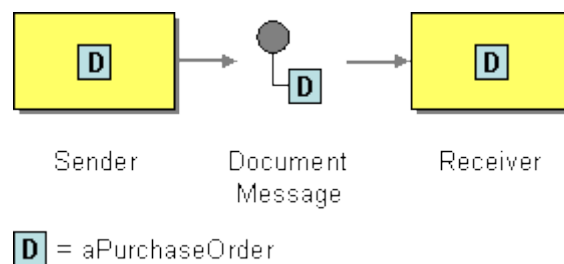


Abbildung 6: Document Message [Abb6]

Das `PersonDocument` stellt in dem Example Daten dar, die aus dem XML-File der Personen ausgelesen wurden und in weiterer Folge in der Datenbank als Kunden abgelegt werden.



#### 4.1.7 Polling Consumer

Bei diesem Pattern ist zum Unterschied zu einigen anderen Patterns der Empfänger für den Aufbau der Verbindung zuständig. Dabei wird in einem bestimmten Intervall abgefragt, ob neue Inhalte zum Abrufen verfügbar sind. Während der Verarbeitung einer Message wird nach keinem neuen Inhalt gefragt. [7]

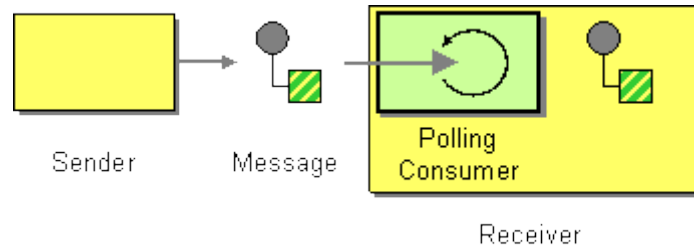


Abbildung 7: Polling Consumer [Abb7]

Wir haben dies bei dem Example überprüft, indem wir während der Laufzeit ein eigenes XML-File einer Person in den Ordner der anderen verfügbaren Files gelegt haben und dieses File im folgenden Intervall abgefragt, transformiert, in die Datenbank abgelegt und in ein neu generiertes XML-File eines Kunden geschrieben wurde.

## 4.2 Beschreibung der Technologien

### 4.2.1 Definition von ETL

„Der Begriff ETL (Extract Transform Load) bezeichnet die drei klassischen Schritte zu Befüllung eines Data Warehouses:

1. *Extraktion:* Die Daten werden aus einem oder mehreren Quellsystemen in das ETL-System geladen.
2. *Transformation:* Die Daten werden im ETL-System umgewandelt. Dabei kann es sich sowohl um technische Umwandlungen wie beispielsweise der Änderung der Datentypen, aber auch um fachliche Umwandlungen wie z.B. von Berechnungen und Aggregationen, handeln.
3. *Load:* Die Daten werden aus dem ETL-System ins Data Warehouse geladen.“ (Klaus Lipinski, ETL (extract transform load), [8])

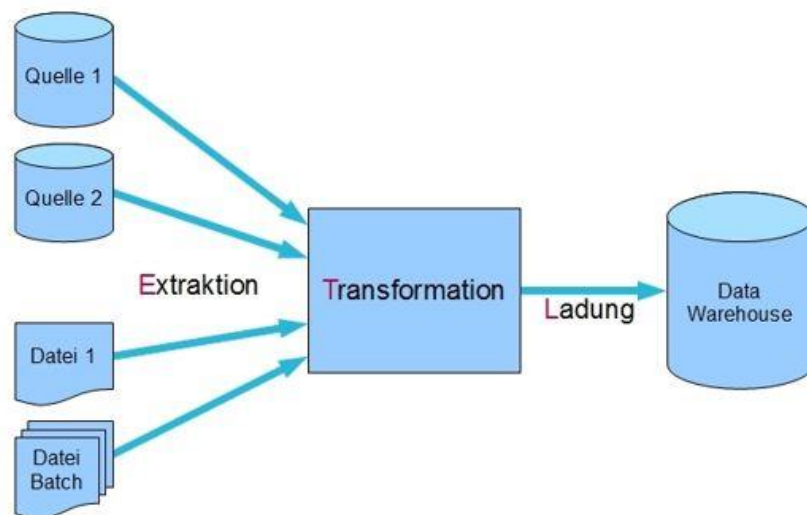


Abbildung 8: ETL-Prozess [Abb8]

### 4.2.2 Definition von EAI

EAI steht für Enterprise Applikation Integration und ist für die Verbindung einzelner Systeme in einer IT-Infrastruktur zuständig. Dabei ist eine Middleware, die als Schnittstelle heterogener Teilnehmer agiert, besonders wichtig, damit diese doch sehr unterschiedlichen Systeme miteinander kommunizieren können. [9]

„Bei  $n$  Softwaresystemen und vollständiger Integration wären bei Punkt-zu-Punkt-Verbindungen im Extremfall  $n * (n-1) / 2$  Verbindungen (und doppelt so viele Schnittstellen) erforderlich. Ein EAI-Bus oder ein Middleware-Layer reduziert die Zahl der notwendigen Verbindungen auf  $n$ . Dadurch reduziert sich der Erstellungs-, Administrations- und Wartungsaufwand.“ (Torsten Horn, EAI Enterprise Application Integration, [9])

#### 4.2.3 Beschreibung und Funktionsweise von Apache Camel

Apache Camel stellt ein von Java zur Verfügung gestelltes Open-Source Routing Framework dar, welches die Integration zwischen mehreren Systemen erleichtert.

Dabei werden verschiedene EIPs (Enterprise Integration Patterns) konkret implementiert. Ebenfalls werden Beispielprogramme zur Verfügung gestellt, mit welchen der Einstieg in die Umgebung erleichtert wird.

Apache Camel stellt eine große Vielfalt von Verbindungsmöglichkeiten zu anderen Systemen, wie z.B. Apache ServiceMix oder Apache ActiveMQ, dar. Die Verbindungen werden in Form von DSLs (Domain Specific Languages) definiert, um die EIPs in einer einfachen Weise zu verkoppeln. [10, 11]

#### Apache Camel's Architektur:

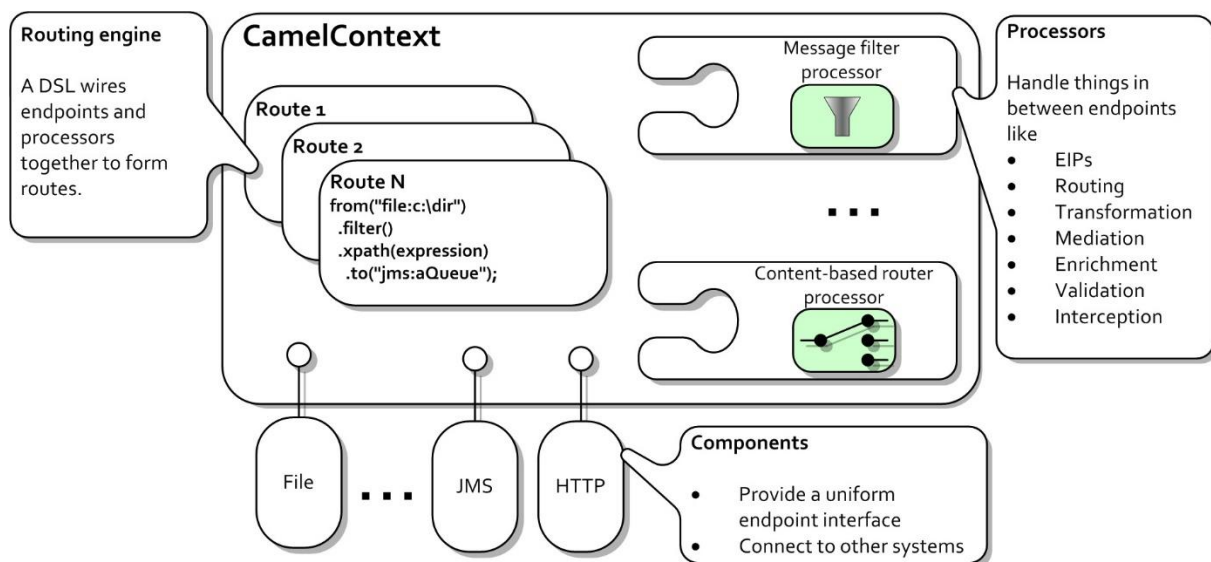


Abbildung 9: Apache Camel's Architektur [Abb9]

Die Architektur besteht aus drei wesentlichen Teilen:

- **Components:** bieten Schnittstellen zu anderen externen Systemen, wie z.B. JMS oder HTTP, um den Core möglichst klein zu halten.
- **Processors:** dienen zum Transformieren und Transportieren der Nachrichten zwischen den Endpunkten. Die 40 EIPs, die Camel momentan unterstützt, sind als Menge von Prozessoren definiert.
- **Routing engine:** werden verwendet, um Endpoints und Prozessoren mittels DSL (Domain Specific Languages), z.B. Java, Scala oder Groovy, miteinander zu koppeln. Dadurch wird eine Route definiert.

[11]

### 4.3 Inbetriebnahme des Beispiels

Folgende Schritte wurden durchgeführt:

1. Binary Zip-Datei von Maven 3.2.5 heruntergeladen: [12]
2. Inhalt der Zip-Datei zu dem Programm-Ordner hinzugefügt
3. Path geändert – mit folgendem Verzeichnis erweitert:  
C:\Program Files (x86)\apache-maven-3.2.5\bin
4. Neue Umgebungsvariable `M2_HOME` hinzugefügt – folgender Wert wurde gesetzt:  
C:\Program Files (x86)\apache-maven-3.2.5
5. Von dem Git-Repository `apache/camel` [13] das ETL-Example, welches im Verzeichnis `examples/camel-example-etl` liegt, heruntergeladen.
6. In Eclipse wurde das Projekt folgendermaßen importiert:  
Rechtsklick auf den Project Explorer -> Import -> Maven -> Existing Maven Projects -> zum Verzeichnis navigieren -> Finish klicken
7. Danach konnte Maven das Projekt folgendermaßen builden:  
Rechtsklick auf das importierte Projekt -> Run as -> Maven build
8. Nachdem das Projekt gebuildet wurde, wurde der Befehl `mvn compile` über die CMD ausgeführt.
9. Das Programm konnte schlussendlich mit dem Befehl `mvn camel:run`, welcher ebenfalls über die CMD erfolgte, gestartet werden und die entsprechenden XML-Files für die Kunden generiert. Folgende Ausgabe erschien in der Konsole:

Hier wird eine neue Customer Entity aufgrund des XML-Files der Person erstellt.

```
2278 camel TRACE [Camel (camel) thread #0 - file:///src/data] openjpa.jdbc.SQL - <t 1
893114627, conn 949303011> [0 ms] spent
2015-02-20 12:44:00,017 [file:///src/data] INFO CustomerTransformer - Found
a matching CustomerEntity Customer[userName: hiram firstName: Hiram surname: Chirino] h
aving the userName hiram.
2015-02-20 12:44:00,017 [file:///src/data] INFO CustomerTransformer - Create
d object customer: Customer[userName: hiram firstName: Hiram surname: Chirino]
```

Die ermittelten Customer-Werte werden in die Datenbank geschrieben.

```
9908 camel TRACE [Camel (camel) thread #0 - file:///src/data] openjpa.jdbc.SQL - <t 8
99471817, conn 521648893> [0 ms] spent
9909 camel TRACE [Camel (camel) thread #0 - file:///src/data] openjpa.jdbc.SQL - <t 8
99471817, conn 881066462> executing prepstmnt 235015338
INSERT INTO customer (id, city, firstName, phone, street, surname,
                      userName, zip)
VALUES (?, ?, ?, ?, ?, ?, ?, ?)
[params=?, ?, ?, ?, ?, ?, ?, ?]
```

Es wird ein SELECT-Befehl ausgeführt, bei dem die Werte der Customer ausgelesen werden.

```
6712 camel TRACE [Camel (camel) thread #1 - jpa://org.apache.camel.example.etl.CustomerEntity] openjpa.jdbc.SQL - <t 1777822419, conn 1217426150> executing prestmt 910172770
SELECT t0.id, t0.city, t0.firstName, t0.phone, t0.street, t0.surname,
       t0.userName, t0.zip
FROM customer t0
```

Aufgrund der definierten Route werden mit dem SELECT-Befehl die Customer aus der Datenbank gelesen.
















```
6713 camel TRACE [Camel (camel) thread #1 - jpa://org.apache.camel.example.etl.CustomerEntity] openjpa.jdbc.SQL - <t 1777822419, conn 1217426150> [0 ms] spent
2015-02-20 12:44:04,456 [.CustomerEntity] INFO Tracer - ID-MK04-10084-1424432637357-0-10 >>> (route2) from(jpa://org.apache.camel.example.etl.CustomerEntity?consumeDelete=false&consumeLockEntity=false&consumer.initialDelay=3000&delay=3000) --> setHeader[CamelFileName] <<< Pattern:InOnly, Headers:{CamelEntityManager=org.apache.openjpa.persistence.EntityManagerImpl@77c554b, breadcrumbId=ID-MK04-10084-1424432637357-0-9}, BodyType:org.apache.camel.example.etl.CustomerEntity, Body:<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<customer id="1">
  <userName>james</userName>
  <firstName>James</firstName>
  <surname>Strachan</surname>
  <city>London</city>
</customer>
```

Die herausgelesenen Datensätze der Kunden werden in neu generierte XML-Files geschrieben.

```
2015-02-20 12:44:04,463 [.CustomerEntity] INFO Tracer - ID-MK04-10084-1424432637357-0-10 >>> (route2) setHeader[CamelFileName] --> file://target/customers <<< Pattern:InOnly, Headers:{CamelEntityManager=org.apache.openjpa.persistence.EntityManagerImpl@77c554b, CamelFileName=james.xml, breadcrumbId=ID-MK04-10084-1424432637357-0-9}, BodyType:org.apache.camel.example.etl.CustomerEntity, Body:<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<customer id="1">
  <userName>james</userName>
  <firstName>James</firstName>
  <surname>Strachan</surname>
  <city>London</city>
</customer>
```

10. Das Programm kann jederzeit mit den Tastenkombinationen Strg + C abgebrochen werden.

## 5. Testbericht

CustomerEntityTest		
	testSetGetCity	0,023
	testSetGetUserName	0,002
	testSetGetFirstName	0
	testToString	0,001
	testSetGetPhone	0
	testSetGetStreet	0
	testSetGetZip	0
	testSetGetSurname	0,002
	testSetGetId	0
CustomerTransformerTest		
	testToCostumer	0,002
PersonDocumentTest		
	testSetGetCity	0,001
	testSetGetUser	0
	testSetGetFirstName	0,001
	testToString	0
	testSetGetLastName	0,001

Nähere Informationen erhaltest du unter „<target/site/testreport/surefire-report.html>“.



## 6. Lessons learned

- besseres Verständnis der ETL, EAI und EIPs
- genauere Betrachtung der wesentlichen EIPs
- Auseinandersetzung mit Apache Camel
- Interpretation von fremden, nicht dokumentierten Source-Code
- richtiges Zitieren, vor allem von direkten Zitat

## 7. Quellenangaben

- [1] Gregor Hohpe, Bobby Woolf (2003). File Transfer [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/FileTransferIntegration.html> [zuletzt abgerufen am 20.02.2015]
- [2] Gregor Hohpe, Bobby Woolf (2003). Shared Database [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/SharedDataBaseIntegration.html> [zuletzt abgerufen am 20.02.2015]
- [3] Gregor Hohpe, Bobby Woolf (2003). Pipes and Filters [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/PipesAndFilters.html> [zuletzt abgerufen am 20.02.2015]
- [4] Gregor Hohpe, Bobby Woolf (2003). Message Translator [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/MessageTranslator.html> [zuletzt abgerufen am 20.02.2015]
- [5] Gregor Hohpe, Bobby Woolf (2003). Message Endpoint [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/MessageEndpoint.html> [zuletzt abgerufen am 20.02.2015]
- [6] Gregor Hohpe, Bobby Woolf (2003). Document Message [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/DocumentMessage.html> [zuletzt abgerufen am 20.02.2015]
- [7] Gregor Hohpe, Bobby Woolf (2003). Polling Consumer [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/PollingConsumer.html> [zuletzt abgerufen am 20.02.2015]
- [8] Klaus Lipinski (2015). ETL (extract transform load) [Online]. Available at: <http://www.itwissen.info/definition/lexikon/ETL-extract-transfer-load.html> [zuletzt abgerufen am 22.02.2015]
- [9] Torsten Horn (2002, 2007). EAI Enterprise Application Integration [Online]. Available at: <http://www.torsten-horn.de/techdocs/eai.htm> [zuletzt abgerufen am 22.02.2015]
- [10] The Apache Software Foundation (2004, 2014). What is Camel [Online]. Available at: <http://camel.apache.org/what-is-camel.html> [zuletzt abgerufen am 22.02.2015]

- [11] Jonathan Anstey (2011). Open Source Integration with Apache Camel and How Fuse IDE Can Help [Online]. Available at: <http://java.dzone.com/articles/open-source-integration-apache> [zuletzt abgerufen am 22.02.2015]
- [12] The Apache Software Foundation (December 2014). Download Apache Maven 3.2.5 [Online]. Available at: <http://maven.apache.org/download.cgi> [zuletzt abgerufen am 19.02.2015]
- [13] Claus Ibsen (2007, 2015). Github-Repository „camel“ [Online]. Available at: <https://github.com/apache/camel> [zuletzt abgerufen am 19.02.2015]

## 8. Abbildungsverzeichnis

- [Abb1] Gregor Hohpe, Bobby Woolf (2003). File Transfer [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/FileTransferIntegration.html> [zuletzt abgerufen am 20.02.2015]
- [Abb2] Gregor Hohpe, Bobby Woolf (2003). Shared Database [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/SharedDataBaseIntegration.html> [zuletzt abgerufen am 20.02.2015]
- [Abb3] Gregor Hohpe, Bobby Woolf (2003). Pipes and Filters [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/PipesAndFilters.html> [zuletzt abgerufen am 20.02.2015]
- [Abb4] Gregor Hohpe, Bobby Woolf (2003). Message Translator [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/MessageTranslator.html> [zuletzt abgerufen am 20.02.2015]
- [Abb5] Gregor Hohpe, Bobby Woolf (2003). Message Endpoint [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/MessageEndpoint.html> [zuletzt abgerufen am 20.02.2015]
- [Abb6] Gregor Hohpe, Bobby Woolf (2003). Document Message [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/DocumentMessage.html> [zuletzt abgerufen am 20.02.2015]
- [Abb7] Gregor Hohpe, Bobby Woolf (2003). Polling Consumer [Online]. Available at: <http://www.enterpriseintegrationpatterns.com/PollingConsumer.html> [zuletzt abgerufen am 20.02.2015]
- [Abb8] Klaus Lipinski (2015). ETL (extract transform load) [Online]. Available at: <http://www.itwissen.info/definition/lexikon/ETL-extract-transfer-load.html> [zuletzt abgerufen am 22.02.2015]
- [Abb9] Claus Ibsen, Jonathan Anstey (December 2010). Camel in Action. ISBN: 9781935182368 [zuletzt abgerufen am 22.02.2015]