

ETL WITH EAI

SYT / DEZSYS09



20. Februar 2015

Erceg, Kritzl

4AHITT

Inhalt

[1. Aufgabenstellung 2](#_Toc412213239)

[2. Requirementanalyse mit Aufwandabschätzung 3](#_Toc412213240)

[3. anschließende Endzeitaufteilung 4](#_Toc412213241)

[3.1 Erceg 4](#_Toc412213242)

[3.2 Kritzl 4](#_Toc412213243)

[3.3 Gesamtsumme 4](#_Toc412213244)

[4. Arbeitsdurchführung 5](#_Toc412213245)

[4.1 Identifikation und Beschreibung der EAI Patterns 5](#_Toc412213246)

[4.1.1 File Transfer 5](#_Toc412213247)

[4.1.2 Shared Database 5](#_Toc412213248)

[4.1.3 Pipes and Filters 6](#_Toc412213249)

[4.1.4 Message Translator 6](#_Toc412213250)

[4.1.5 Message Endpoint 6](#_Toc412213251)

[4.1.6 Document Message 6](#_Toc412213252)

[4.1.7 Polling Consumer 7](#_Toc412213253)

[4.2 Beschreibung der Funktionsweise von Apache Camel 8](#_Toc412213254)

[4.3 Inbetriebnahme des Beispiels 9](#_Toc412213255)

[5. Testbericht 11](#_Toc412213256)

[6. Lessons learned 12](#_Toc412213257)

[7. Quellenangaben 13](#_Toc412213258)

[8. Abbildungsverzeichnis 14](#_Toc412213259)

*Github-Link:* [*https://github.com/mkritzl-tgm/DezSys09-EAI*](https://github.com/mkritzl-tgm/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 verstehen | Erceg, Kritzl | 50 min |  |
| Code dokumentieren | Erceg, Kritzl | 100 min | x |
| Herausfinden, welche Enterprise Integration Patterns im Example verwendet wurden | Erceg, Kritzl | 120 min |  |
| **Testung des Programms** |  |  |  |
| Unit-Tests |  |  |  |
| Integration-Tests |  |  |  |
| System-Tests |  |  |  |
| User-Acceptance-Tests |  |  |  |

# 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 |
|  |  |  |
| *Gesamt* | *20.02.2015* | **Minuten (h min)** |

## 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 |
|  |  |  |
| *Gesamt* | *20.02.2015* | **Minuten (h min)** |

## 3.3 Gesamtsumme

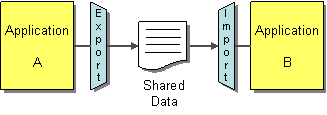
Insgesamt haben wir für diese Übung **Stunden und Minuten** benötigt. Geschätzt wurden 24 Stunden, daher lag unsere Einschätzung ziemlich daneben.

# 4. Arbeitsdurchführung

## 4.1 Identifikation und Beschreibung der EAI Patterns

### 4.1.1 File Transfer

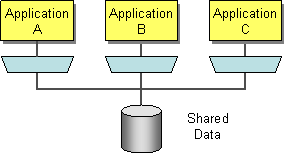
Damit mehrere Applikationen miteinander agieren können und die jeweiligen Informationen austauschen können, wird als Pattern das File Transfer angeboten. Dabei stellen eine oder mehrere Dateien den zentralen Mittelpunkt des Patterns dar, die in einem bestimmten Intervall erneuert werden. „Integrators“ 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

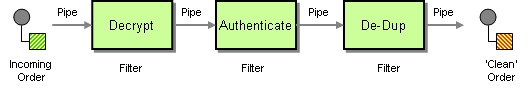


*Abbildung 2: Shared Database [Abb2]*

Im ETL-Example kommt das Pattern bei der Transferierung der XML-Files zu POJO (Plain Old Java Object) und in weiterer Folge in das JPA (Java Persistence API). Dies stellt die Export-Seite dar. Die Import-Seite ist für das Herauslesen der Kundeninformationen aus der Datenbank und das Erstellen der XML-Files für die Kunden zuständig.

### 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]

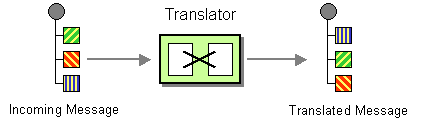


*Abbildung 3: Pipes and Filters [Abb3]*

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

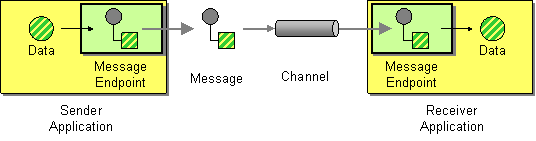
Bei diesem Pattern wird eine ankommende Nachricht vom Translator mittels eines Filters in eine andere Form gebracht. Dies kommt zu Stande, wenn der Sender und der Empfänger nicht die gleichen Interfaces für die Kommunikation verwenden und dadurch die gesendete Nachricht vom Empfänger nicht verstanden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



*Abbildung 4: Message Translator [Abb4]*

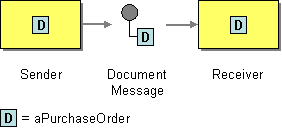
Der Customer Translator stellt in dem Example den Message Translator dar und übersetzt die Informationen der Person zu Informationen des Kunden.

### 4.1.5 Message Endpoint



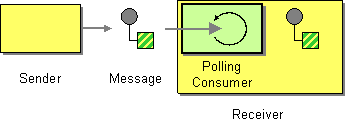
*Abbildung 5: Message Endpoint [Abb5]*

### 4.1.6 Document Message



*Abbildung 6: Document Message [Abb6]*

### 4.1.7 Polling Consumer



*Abbildung 7: Polling Consumer [Abb7]*

## 4.2 Beschreibung der Funktionsweise von Apache Camel

## 4.3 Inbetriebnahme des Beispiels

Folgende Schritte wurden durchgeführt:

1. Binary Zip-Datei von Maven 3.2.5 heruntergeladen: [8]
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 [9] 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:

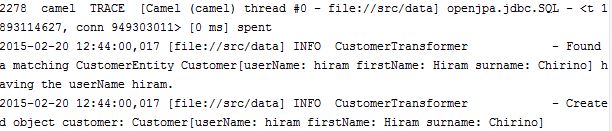
Rechtsklick auf das importierte Projekt -> Run as -> Maven build

8. Nachdem das Projekt gebuildet wurde, wurde der Befehl mvn compile über die CMD

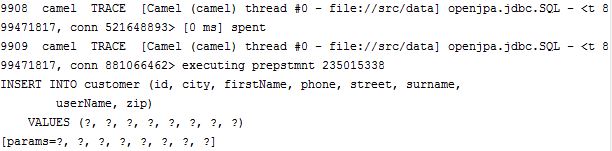
ausgeführt.

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:

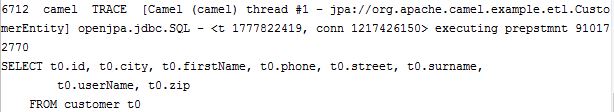
Hier wird eine neue Customer Entity aufgrund des XML-Files der Person erstellt.



Die ermittelten Customer-Werte werden in die Datenbank geschrieben.

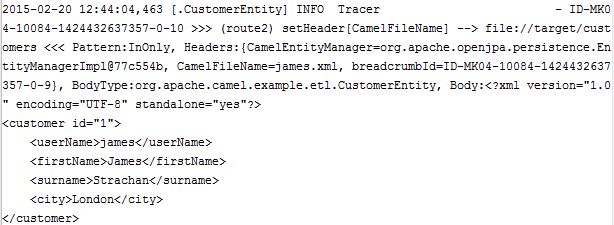


Es wird ein SELECT-Befehl ausgeführt, bei dem die Werte der Costumer ausgelesen werden.



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

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



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

# 5. Testbericht

# 6. Lessons learned

# 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] 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]

[9] 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]