2014/15

ETL mit EAI

TGM WIEN xx | 22.02.2015

4BHIT

Adler/Karic

Inhaltsverzeichnis

[1. Identifikation und Beschreibung der EAI Patterns 2](#_Toc412387886)

[2. Funktionsweise von Apache Camel 2](#_Toc412387887)

[3. Inbetriebnahme des Beispiels 2](#_Toc412387888)

[4. Literaturverzeichnis 4](#_Toc412387889)

# Identifikation und Beschreibung der EAI Patterns

<http://www.eaipatterns.com/docs/jaoo_hohpeg_enterpriseintegrationpatterns.pdf>

🡪 besser erklärt

<http://de.slideshare.net/ieugen222/eip-cu-apache-camel>

(aufpassen, nur die patterns die in dem Beispiel vorkommen, schau halt dass du die wirklich beschreibst.. vieles einfach kopieren..)

<http://www.ossmentor.com/2014/11/camel-and-enterprise-integration.html>

# 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.[2] Darüber hinaus wird oft auch Apache MINA (Framework für Netzwerkapplikationen) zusammen mit Apache Camel verwendet.

<http://help.eclipse.org/kepler/index.jsp?topic=%2Forg.eclipse.stardust.docs.camel%2Fhtml%2Fcamel%2Fcamel-introduction.html>

<https://www.youtube.com/watch?v=z6TizMFTIvI>

# Inbetriebnahme des Beispiels

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





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)

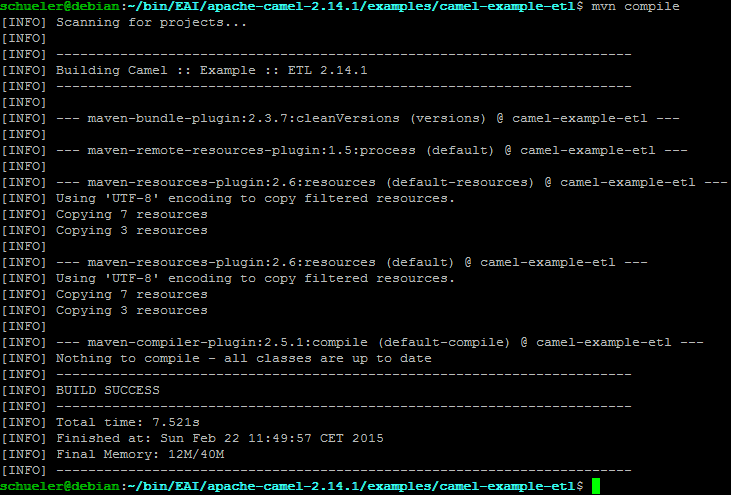


Wie wir im README lesen können müssen wir um das Beispiel zu starten zwei Befehle ausführen:

*mvn compile*

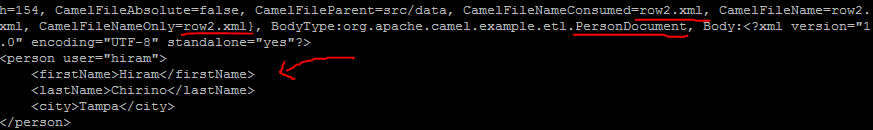
*mvn camel:run*

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:

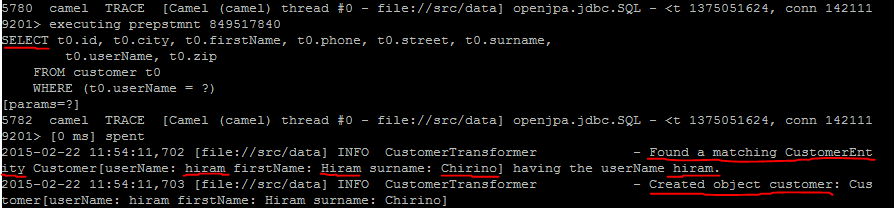


Wir erhalten ein BUILD SUCCESS und führen nun den Befehl *mvn camel:run* aus:

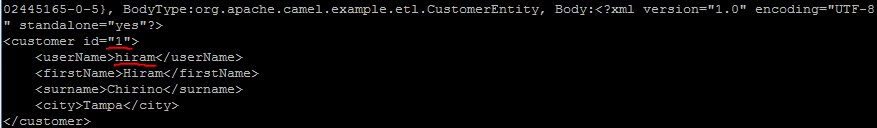
Wir sehen zunächst (nach mehreren Infos) die Ausgabe einer Person (als XML-Struktur). Man sieht auch von wo diese Daten kommen: row2.xml und wie sie intern repräsentiert werden nämlich als PersonDocument.



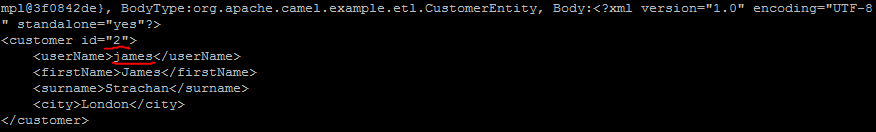
Danach folgt ein SELECT aus der Datenbank und anschließend findet der CustomerTransformer eine passende CustomerEntity (hiram). Zuletzt wird noch das Customer Objekt erzeugt. Derselbe Prozess passiert auch bei der 2.Person (James)



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



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



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

# Literaturverzeichnis

Wikipedia angeben? (funktweise)

**There are no sources in the current document.**