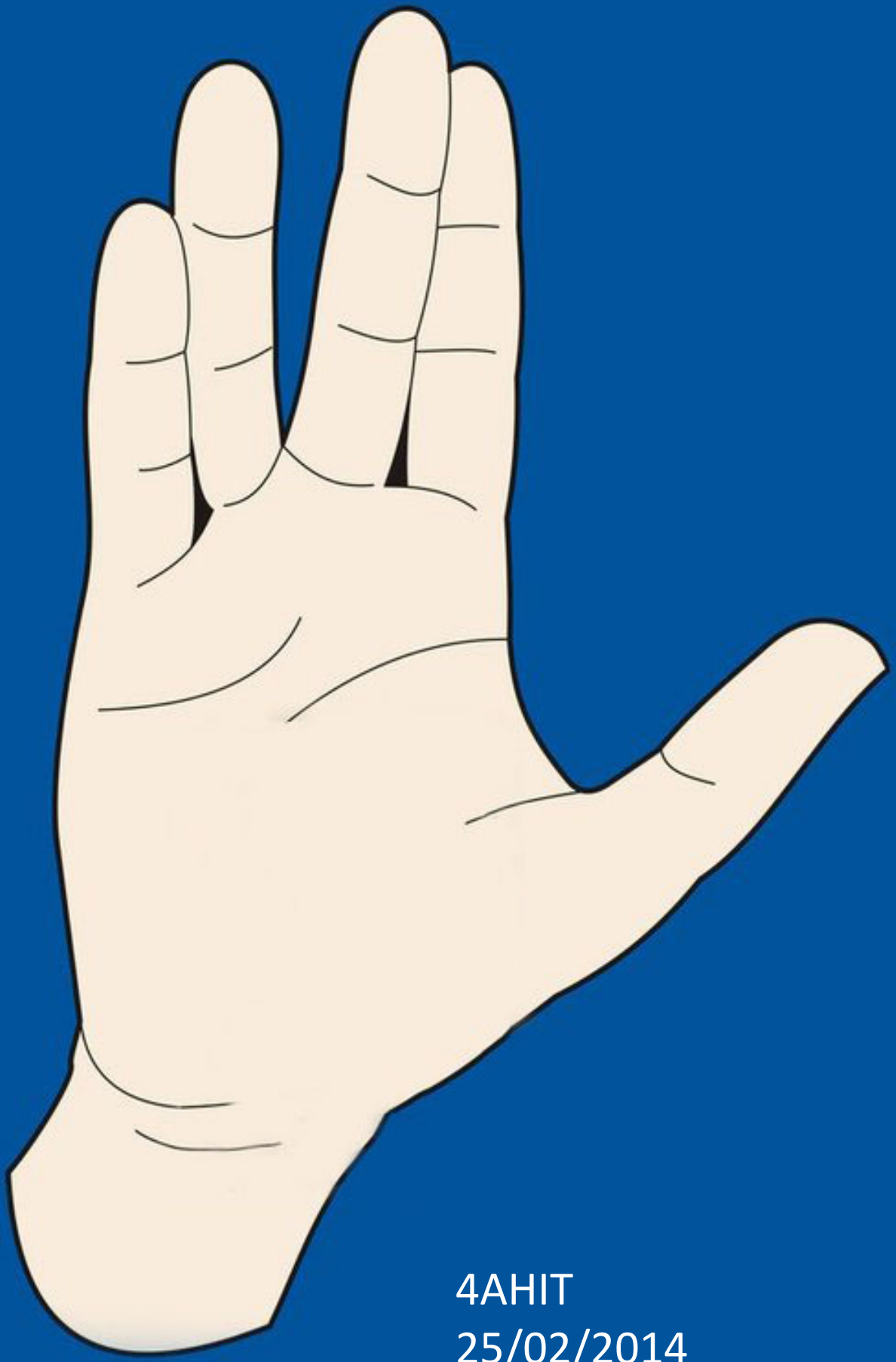


ETL - EAI

4AHIT
25/02/2014
Christian Janeczek
Wolfgang Mair

Inhaltsverzeichnis

Aufgabenstellung

ENTERPRISE APPLICATION INTEGRATION

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 13.02.2014
[2] Enterprise Integration Patterns; G.Hohpe, B.Woolf; 2003; Online: http://www.enterpriseintegrationpatterns.com/toc.html; abgerufen 13.02.2014
[3] Extract Transform Load (ETL) Example; Apache Camel; Online: http://camel.apache.org/etl-example.html; abgerufen 13.02.2014

Designüberlegung

IMPLEMENTIERUNG DES FRAMEWORKS APACHE CAMEL:

- Installation des Services Apache Camel

TESTING DER FUNKTIONALITÄT ANHAND DES BEISPIELS ETL:

- Das Example mit dem Namen „ETL“ zum Laufen bringen

ANALYSE DER ENTERPRISE INTEGRATION PATTERNS:

- Den Einsatz, der im Beispiel vorkommenden Enterprise Integration Patterns schildern

Technologie

EAI

EAI oder auch Enterprise Application Integration wird als Middleware zur Kopplung von unterschiedlichen IT-Systemen benutzt. EAI kommt meistens bei Betriebswirtschaftlicher Software wie z.B.: CRP, SCM, etc. vor.

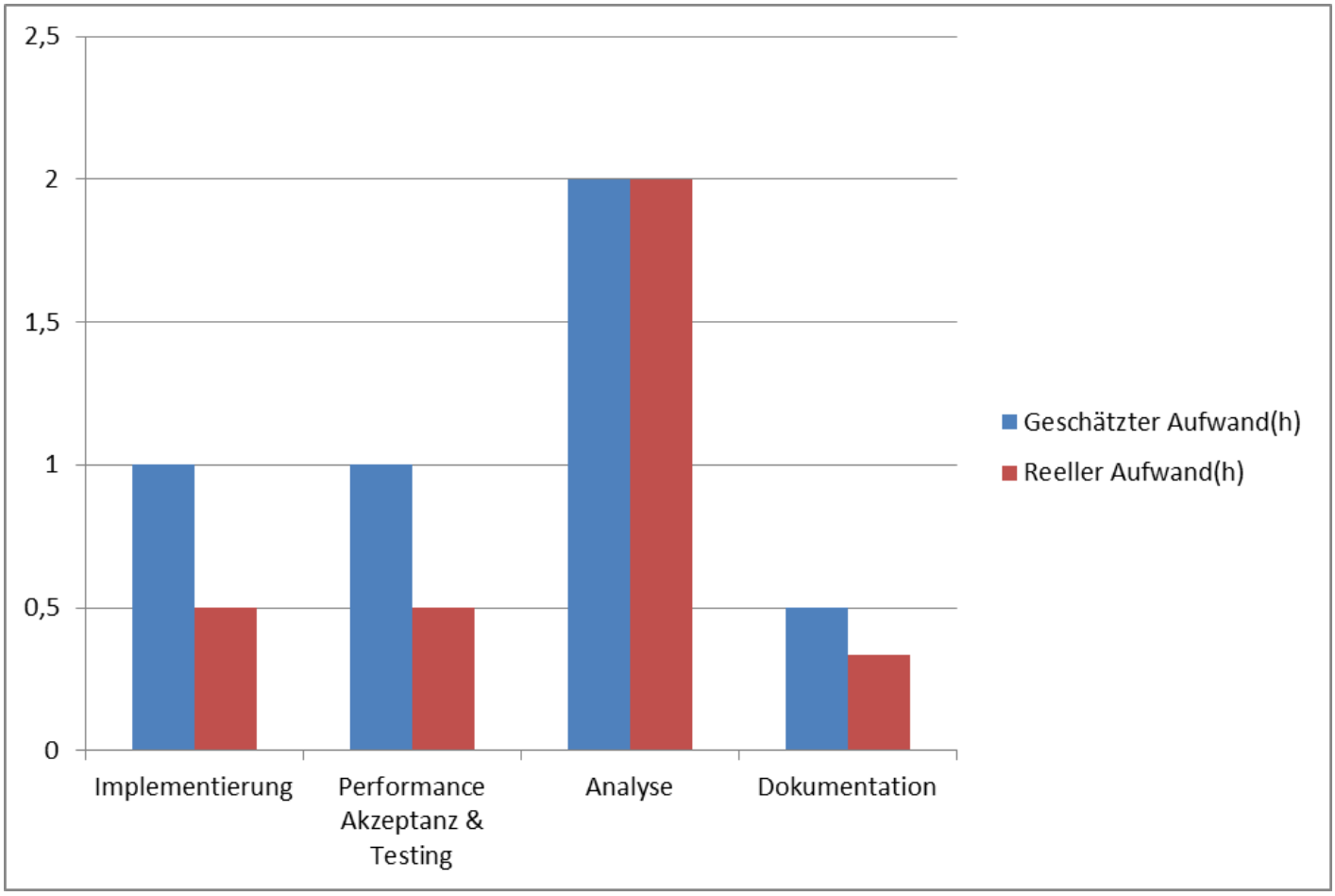
EAI ermöglicht es also möglichst einfach eine hohe Flexibilität der Systemintegration zu erzielen.*[10]*

Aufwandsabschätzung

Kategorie	Beschreibung	Geschätzter Aufwand(h)
Implementierung	Implementierung des Frameworks Apache Camel	1
Performance Akzeptanz & Testing	Testing der Services Apache Camel anhand des Examples "ETL"	1
Analyse	Analyse bezüglich der Benutzung von Enterprise Integration Patterns	2
Dokumentation	Schreiben der Dokumentation	0,5

Kategorie	Kommentar	Reeller Aufwand(h)
Implementierung	Implementierung war relativ einfach	0,5
Performance Akzeptanz & Testing	Ausführen des Examples mit dem Plugin Maven war nicht allzu zeitaufwändig	0,5
Analyse		2
Dokumentation		0,333333333

	Geschätzter Aufwand(h)	Reeller Aufwand(h)
Implementierung	1	0,5
Performance Akzeptanz & Testing	1	0,5
Analyse	2	2
Dokumentation	0,5	0,333333333



Arbeitsdurchführung

Erfolge:

- Implemeniterung sowie Analyse ohne jegliche Problem erfolgreich.

Niederlagen:

- -

BESCHREIBUNG DER EINZELNEN KLASSEN

Main.java

Ruft die run Methode von der Main Klasse auf. Da keine run Methode in der Main Klasse vorhanden ist, gehe ich davon aus das es in der Superklasse org.apache.camel.spring.Main deklariert wird.

CustomerEntity.java

```
@Entity(name = "customer")
@XmlRootElement(name = "customer")
@XmlAccessorType(XmlAccessType.FIELD)
@NamedQuery(name = "findCustomerByUsername", query = "SELECT c FROM customer c WHERE c.userName = :userName")
```

In der Klasse CustomerEntity befindet sich die Funktionalität eine XML Datei daten auszulesen.

```
@XmlAttribute
private Long id;
private String userName;
private String firstName;
private String surname;
```

Währendessen beeinhaltet die Klasse nebenbei auch Attribute und getter und setter dieser Attribute.

PersonDocument.java

```
@XmlRootElement(name = "person")
@XmlAccessorType(XmlAccessType.FIELD)
public class PersonDocument {
    @XmlAttribute
    private String user;
    @XmlElement
    private String firstName;
    @XmlElement
    private String lastName;
    @XmlElement
    private String city;
```

In der Klasse PersonDocument werden Java Attribute mit XML kombiniert und definiert. Desweiteren beinhaltet die Klasse Setter und Getter für diese Attribute.

CustomerTransformer.java

CustomerTransformer besitzt eine @Converter Methode. Die Converter Methode wandelt die Daten dann weiter damit es in einer Datenbank eingetragen werden kann.

EtlRoutes.java

```
from("file:src/data?noop=true")
    .convertBodyTo(PersonDocument.class)
    .to("jpa:org.apache.camel.example.etl.CustomerEntity");
```

Die EtlRoutes Klasse versucht das PersonDocument in eine CustomerEntity mittels TypeConverter zu transformieren.

```
from("jpa:org.apache.camel.example.etl.CustomerEntity?consumeDelete=false&delay=3000&consumeLockEntity=false")
    .setHeader(Exchange.FILE_NAME, el("${in.body.userName}.xml"))
    .to("file:target/customers");
```

Und dann auch weiter die Entity in eine .xml Datei umzuwandeln, welches dann in target/customers gespeichert wird.

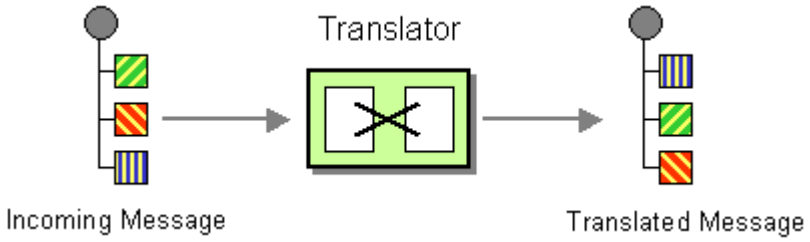
Allgemein

Im Grunde liest das Programm aus einer XML Datei aus wandelt diese dann mittels eines TypeConverters in eine PersonDocument Klasse um, welche dann mittels eines selbst definierten Transformers weiter in einen CustomerEntity transformiert wird. Danach wird es noch weiter transformiert um es in eine Datenbank zu speichern.

ANALYSE DER ENTERPRISE INTEGRATION PATTERNS:

Messaging Systems:

- **Message Translator**, How can systems using different data formats communicate with each other using messaging?



„Camel supports the **Message Translator** from the **EIP patterns** by using an arbitrary **Processor** in the routing logic, by using a **bean** to perform the transformation, or by using transform() in the DSL. You can also use a **Data Format** to marshal and unmarshal messages in different encodings.“[2]

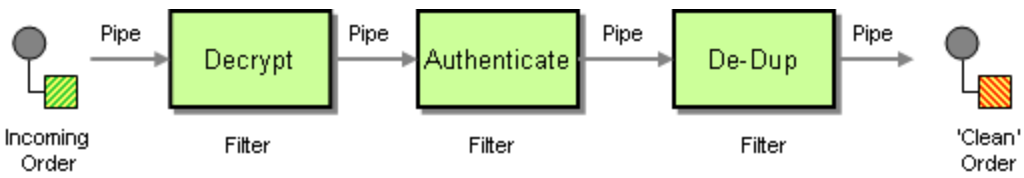
Bei dem Beispiel „ETL“ handelt es sich bei der Klasse CustomerTransformer um das Pattern eines Messaging Systems.

Der CustomerTransformer übernimmt also die Rolle des Message Translators.

Er liest sozusagen die Daten aus einem **XML**-File aus, erstellt daraus ein personales Dokument(**PersonDocument**) und fasst daraus eine **CustomerEntity** zusammen.

„Use a special filter, a *Message Translator*, between other filters or applications to translate one data format into another.“[1]

- **Pipes and Filters,** How can we perform complex processing on a message while maintaining independence and flexibility?

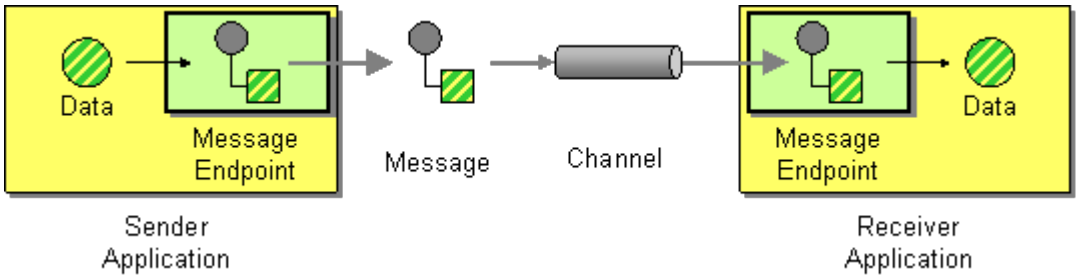


„Camel supports the **Pipes and Filters** from the **EIP patterns** in various ways.

With Camel you can split your processing across multiple independent **Endpoint** instances which can then be chained together.“[3]

„Use the *Pipes and Filters* architectural style to divide a larger processing task into a sequence of smaller, independent processing steps (Filters) that are connected by channels (Pipes).“[4]

- **Message Endpoint,** How does an application connect to a messaging channel to send and receive messages?



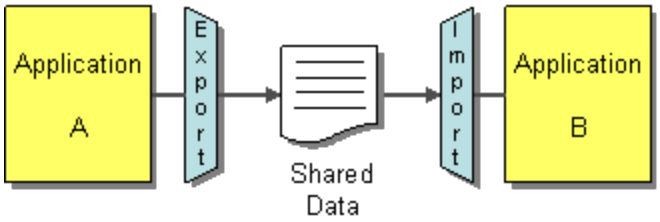
„Camel supports the **Message Endpoint** from the **EIP patterns** using the **Endpoint** interface.
When using the **DSL** to create **Routes** you typically refer to Message Endpoints by their **URIs** rather than directly using the **Endpoint** interface. Its then a responsibility of the **CamelContext** to create and activate the necessary Endpoint instances using the available **Component** implementations.“[5]

Im Beispiel „ETL“ findet die Definition der Endpoints mittels der Klasse **EtlRoutes** statt.

Verzeichnis: **src/data** .convertBodyTo(PersonDocument.class)

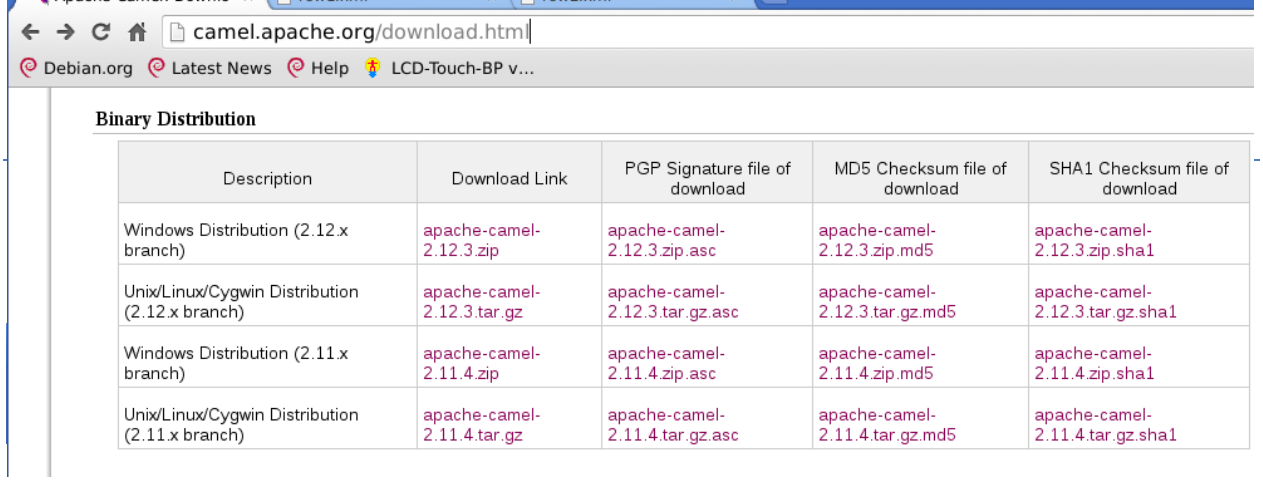
JPA Endpoint: **.to("jpa:org.apache.camel.example.etl.CustomerEntity");**

- **File Transfer,** How can I integrate multiple applications so that they work together and can exchange information?



„Have each application produce files containing information that other applications need to consume. Integrators take the responsibility of transforming files into different formats. Produce the files at regular intervals according to the nature of the business.“[6]

Bei unserem Beispiel „ETL“ werden Daten aus XML-Files extrahiert und in andere Dateiformate transferiert(Plain Old Java Object, JPA).



Binary Distribution

Description	Download Link	PGP Signature file of download	MD5 Checksum file of download	SHA1 Checksum file of download
Windows Distribution (2.12.x branch)	apache-camel-2.12.3.zip	apache-camel-2.12.3.zip.asc	apache-camel-2.12.3.zip.md5	apache-camel-2.12.3.zip.sha1
Unix/Linux/Cygwin Distribution (2.12.x branch)	apache-camel-2.12.3.tar.gz	apache-camel-2.12.3.tar.gz.asc	apache-camel-2.12.3.tar.gz.md5	apache-camel-2.12.3.tar.gz.sha1
Windows Distribution (2.11.x branch)	apache-camel-2.11.4.zip	apache-camel-2.11.4.zip.asc	apache-camel-2.11.4.zip.md5	apache-camel-2.11.4.zip.sha1
Unix/Linux/Cygwin Distribution (2.11.x branch)	apache-camel-2.11.4.tar.gz	apache-camel-2.11.4.tar.gz.asc	apache-camel-2.11.4.tar.gz.md5	apache-camel-2.11.4.tar.gz.sha1

Downloaden von Apache Camel welches auf der Webseite von Apache Camel zu finden ist. Einfach auf der Hauptseite von Apache Camel auf Download klicken welches den Link http://camel.apache.org/download.html aufruft.

Binary Distribution

Description	Download Link	PGP Signature file of download	MD5 Checksum file of download	SHA1 Checksum file of download
Windows Distribution (2.12.x branch)	apache-camel-2.12.3.zip	apache-camel-2.12.3.zip.asc	apache-camel-2.12.3.zip.md5	apache-camel-2.12.3.zip.sha1
Unix/Linux/Cygwin Distribution (2.12.x branch)	apache-camel-2.12.3.tar.gz	apache-camel-2.12.3.tar.gz.asc	apache-camel-2.12.3.tar.gz.md5	apache-camel-2.12.3.tar.gz.sha1

In unserem Fall benötigen wir Debian da wir es auf der Virtuellen Instanz installiert und getestet haben. Also den Download-Link von Debian (Hier gelb hervorgehoben) anklicken.

```
schueler@debian:~/camel$ gunzip apache-camel-2.12.3.tar.gz
schueler@debian:~/camel$ tar -xfv apache-camel-2.12.3.tar
```

Entpacken der gerade gedownloadeten Dateien mittels den oben angezeigten Befehlen.

```
schueler@debian:~/camel$ sudo apt-get install maven
```

Nach dem entpacken müssen wir Maven mittels apt-get install downloaden und installieren.

```
apache-camel-2.12.3/examples/camel-example-etl
```

Um das vorgefertigte Beispiel nun auszuführen gehen wir in den oben genannten Pfad und lesen die Readme hier lesen wir folgendes:

```
You will need to compile this example first:
  mvn compile

To run the example type
  mvn camel:run
```

Also tun wir beides und was dann erscheint ist folgendes (siehe Testbericht)

Testbericht

Der Outcome von der Durchführung des Examples:

```
schueler@debian:~/camel/apache-camel-2.12.3/examples/camel-example-etl$ mvn cam
el:run
```

Die Eingabe des Befehls zum starten des Beispiels.

```
INFO]
INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ c
mel-example-etl ---
INFO] Using 'UTF-8' encoding to copy filtered resources.
INFO] skip non existing resourceDirectory /home/schueler/camel/apache-camel-2.
2.3/examples/camel-example-etl/src/test/resources
INFO] Copying 3 resources
INFO]
INFO] --- maven-compiler-plugin:2.5.1:testCompile (default-testCompile) @ came
-example-etl ---
INFO] Nothing to compile - all classes are up to date
INFO]
INFO] <<< camel-maven-plugin:2.12.3:run (default-cli) @ camel-example-etl <<<
INFO]
INFO] --- camel-maven-plugin:2.12.3:run (default-cli) @ camel-example-etl ---
```

Als erstes kommen eineige Infos und Downloads.

```
2014-02-27 18:49:53,508 [.CustomerEntity] INFO  Tracer
- ID-debian-45712-1393523329957-0-72 >>> (route2) setHeader[CamelFileName] -->
file://target/customers <<< Pattern:InOnly, Headers:{CamelFileName=james.xml, C
amelEntityManager=org.apache.openjpa.persistence.EntityManagerImpl@25662980, br
eadcrumbId=ID-debian-45712-1393523329957-0-71}, BodyType:org.apache.camel.examp
le.etl.CustomerEntity, Body:<?xml version="1.0" encoding="UTF-8" standalone="ye
s"?>
<customer id="2">
    <userName>james</userName>
    <firstName>James</firstName>
    <surname>Strachan</surname>
    <city>London</city>
</customer>
```

Danach werden die Relevanten Messages ausgegeben.

Das Example kann man nur mittels Strg+C stoppen.

Quellenangabe

ID	Titel	Autor/Institution	Erstellungs-/Aktualisierungsdatum	Link	zuletzt abgerufen am
10	EAI Enterprise Application Integration	Torsten Horn	Freitag, 15. April 2011	http://www.torsten-horn.de/techdocs/eai.htm	27.02.2014
1	Message Translator	Gregor Hohpe and Bobby Woolf	Montag, 25. Mai 2009	http://www.enterpriseintegrationpatterns.com/MessageTranslator.html	27.02.2014
2	Message Translator Description	Gregor Hohpe and Bobby Woolf	Dienstag, 11. Februar 2014	https://camel.apache.org/message-translator.html	27.02.2014
3	Pipes and Filters	Gregor Hohpe and Bobby Woolf	Dienstag, 11. Februar 2014	https://camel.apache.org/pipes-and-filters.html	27.02.2014
4	Pipes and Filters	Gregor Hohpe and Bobby Woolf	Montag, 25. Mai 2009	http://www.enterpriseintegrationpatterns.com/PipesAndFilters.html	27.02.2014
5	Message Endpoint	Gregor Hohpe and Bobby Woolf	Montag, 25. Mai 2009	http://www.enterpriseintegrationpatterns.com/MessageEndpoint.html	27.02.2014
6	File Transfer	Gregor Hohpe and Bobby Woolf	Montag, 25. Mai 2009	http://www.eaipatterns.com/FileTransferIntegration.html	27.02.2014