Masarykova univerzita
Fakulta informatiky


# RESTEasy integration with Apache Camel project

Master's thesis

**Roman Jakubčo**

Brno, Spring 2015

# Declaration

Hereby I declare, that this paper is my original authorial work, which I have worked out by my own. All sources, references and literature used or excerpted during elaboration of this work are properly cited and listed in complete reference to the due source.

Roman Jakubčo

**Advisor:** Mgr. Marek Grác, PhD.

# Acknowledgement

# Abstract

## Keywords

# Contents

# 1 Introduction

# 2 Technologies

This chapter describes technologies that are used for implementation of Camel RESTEasy component. Each subsection introduces the technology and tries to explain what is its main functionality and common usage.

## 2.1 Apache Camel

Apache Camel is a open source rule-based routing and mediation framework implemented in Java. It is based on theory of Enterprise Integration Patterns or EIP, described in the book with same name written by Gregor Hohpe and Bobby Wolf[1].

Its main focus is on integration and interaction between various applications or systems for which Camel can provide standalone routing, transformation, monitoring and many other things. From this point of view Camel may seems like ESB[1], but this is not the case, because Camel doesn't provide a container support or reliable message bus, but it can be deployed into one and create full integration platforms(also know as ESB) like Apache ServiceMix[2], JBoss Fuse[3] and JBoss Fuse Service Works.

Camel also has extensible and modular architecture that allows implementation and seamlessly plug in support for new protocols and this architectural design makes Camel lightweight, fast and easy extendable for developers.[2]

### 2.1.1 The core principles/features of Camel

Camel is using convention over configuration approach to describe given task by domain-specific language (DSL) in declarative way. This way Camel minimize number of lines of the source code that is needed for implementation of integration scenarios. Another key feature that helps with this task is usage of theory of EIPs, which are already integrated in DSL and also getting the most from their potential.

---

1. ESB - Enterprise Service Bus
2. `http://servicemix.apache.org/`
3. `http://www.jboss.org/products/fuse/overview/`

Another fundamental principle of Camel is that it makes no assumptions about the data format. This feature is important because it makes possible for developer to integrate systems together without need to convert data to some canonical format. This way there are no limitations for integration of any kind of systems together[2].

### Routing and mediation engine

One of the core features of Camel is its routing and mediation engine. A routing engine selectively move a message from one destination to another based on the route's configuration. Users also can define their own rules for routing, add processors to modify the content of messages, filter them based on some predicate and at the end decide the final destination for the delivery of the message.

### Enterprise integration patterns

Like it is mentioned before Camel is based primarily on EIPs. EIPs describe integration problems and their solutions and also provide some basic vocabulary but the problem is that this vocabulary isn't formalized. And in this comes Camel with its language which describes the integration solutions and tries to formalized the vocabulary. There's almost a one-to-one relationship between the patterns described in Enterprise Integration Patterns and the Camel DSL[4]. Almost all of the EIPs that are definied in the book are implemented as Processors or sets of Processors in the Camel. Processors are used for manipulation of messages between destinations specified in the Camel route.[3]

### Domain-specific language

There are few other integration frameworks with DSL and also some have support for describing route rules in XML, but bonus that comes with Camel is the support for specifying DSLs in regular programming languages as Java, Groovy, Ruby and even Scala. Of course there is also possibility to describe the route in XML document.

---

4. `http://camel.apache.org/enterprise-integration-patterns.html`

### Modular and pluggable architecture

The next feature is the approach to the architecture, which is done in modular way. This means, that Camel can be easily extended to consume data from endpoint and produce data to endpoint. Camel is describing this as developing a new component, where each component is responsible for consuming or producing data for some specific endpoint and technology e.g. file, HTTP and many others. When developers want to develop a new component for some unique system and add its functionality to the Camel, they just need to follow structure specified by the framework and extend core classes.

By default Camel ships with the few most basic components called camel-core. This bundle includes 24 components including components like bean, file, log, seda and mock. Plus there are many more components developed by the Apache community and also third-parties[5]. There are already developed components that can be used for the most common integration scenarios that occur in systems. Some components worthy of note are web services including SOAP[6] or REST[7], JMS[8], specialized JMS component for Apache ActiveMQ[9] or components for different database connections.

### Configuration

As mentioned before Camel uses convention over configuration paradigm to minimize configuration requirements so that developers don't need to learn complicated configuration options and focus on more important things. This is reflected on configuration of endpoints in route definitions with URI options as can be seen on example **??**.

### Type converters

### Lightweight framework

5. `http://camel.apache.org/components.html`
6. SOAP
7. REST
8. JMS
9. `http://camel.apache.org/activemq.html`

# References

[1] HOHPE, Gregor and WOLF, Bobby. *Enterprise integration patterns*. Boston: Addison-Wesley, c2003, li, ISBN 978-0321200686.

[2] IBSEN, Claus and ANSTEY,Jonathan. *Camel in Action*. Greenwich, Conn.: Manning, c2011, xxxi, ISBN 19-351-8236-6.

[3] http://java.dzone.com/articles/open-source-integration-apache

# A Appendix