





Welcome to

## 8. REST

### KEA System Integration

Henrik Kramselund Jereminsen [hkj@zencurity.com](mailto:hkj@zencurity.com) @kramse  

Slides are available as PDF, [kramse@Github](mailto:kramse@Github)  
8-REST-system-integration.tex in the repo security-courses

# Goals for today



Today's goals:

- Look more at REST
- Repeat some of the slides and exercises
- See how applications can run more independently using REST
- Working with with REST using Elasticsearch mostly

Photo by Thomas Galler on Unsplash

# ime schedule



- 08:30 2x 45 min with 10min break  
Subject REST SOA chapter 7, Camel chapter 10 We will also be doing small exercises with Python and rest
- 10:15 the rest of the day  
Subject REST and exercises  
We will be a couple of larger, more advanced exercises with REST

Repeat some of the previous parts, including the exercises I asked you to perform from the exercises booklet:

3 Getting started with the Elastic Stack 15 min

15 Postman API Client 20 min

More exercises, dive deeper into the Elastic stack

Exercise 5 Use Ansible to install Elastic Stack OR Use Elasticsearch with sample data

Note: Flexible - so you can do them today, or later in the week if it works better with your own planning

# Plan for today



- REST in misc settings
- Examples from programs we have used

## Exercises

- Postman API Client
- Making requests to Elasticsearch

Note: reading for next time includes SOA book, *Chapter 9: Service API and Contract Design with REST Services and Microservices*

# Reading Summary



SOA book chapter 7: Analysis and Modeling with REST Services and Microservices

Camel book chapter 10: RESTful web services

Repeated:

Yes, there is overlap and the same subjects from different angles. Consider the SOA book a theoretical book, the Camel book a proof of concept

# SOA book chapter 7:



## Analysis and Modeling with REST Services and Microservices

### 7.1 REST Service Modeling Process

### 7.2 Additional Considerations

- This chapter provides a detailed step-by-step process for modeling REST service candidates.
- End result is similar to chapter 6 a service composition candidate

### Source:

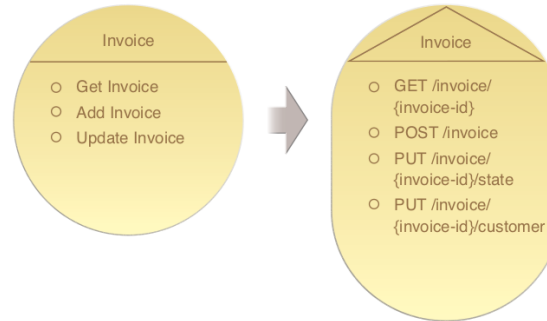
*Service-Oriented Architecture: Analysis and Design for Services and Microservices*, Thomas Erl, 2017

# REST Service Capability Granularity



**Figure 7.19**

A REST service candidate can be modeled specifically to incorporate uniform contract characteristics. The Update Invoice service capability candidate is split into two variations of the PUT /invoice/ service capability: one that updates the invoice state value, and another that updates the invoice customer value.



- REST using HTTP has the standard HTTP methods available (e.g., GET, POST, PUT, DELETE);
- See also [https://en.wikipedia.org/wiki/Representational\\_state\\_transfer](https://en.wikipedia.org/wiki/Representational_state_transfer)

Source:

*Service-Oriented Architecture: Analysis and Design for Services and Microservices*, Thomas Erl, 2017

# Chapter 9: Service API and Contract Design with REST Services and Microservices



On the reading list for next time. More formal about designing APIs

REST service contracts are typically designed around the primary functions of HTTP methods, which make the documentation and expression of REST service contracts distinctly different from operation-based Web service contracts. Regardless of the differences in notation, the same overarching contract-first approach to designing REST service contracts is paramount when building services for a standardized service inventory.

- REST entity service contracts are typically dominated by service capabilities that include inherently idempotent and reliable GET, PUT, or DELETE methods
- This chapter provides service contract design guidance for service candidates modeled as a result of the service-oriented analysis stage covered in Chapter 7.

Source:

*Service-Oriented Architecture: Analysis and Design for Services and Microservices*, Thomas Erl, 2017

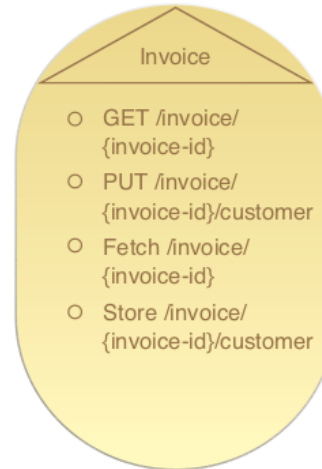


# REST Service



**Figure 9.1**

An entity service based on the Invoice business entity that defines a functional scope that limits the service capabilities to performing invoice-related processing. This agnostic Invoice service will be reused and composed by other services within the same service inventory in support of different automated business processes that need to process invoice-related data. This particular invoice service contract displays two service capabilities based on primitive methods and two service capabilities based on complex methods.



- Very typical REST URL/method GET /invoice/{invoice-id}

Source:

*Service-Oriented Architecture: Analysis and Design for Services and Microservices*, Thomas Erl, 2017

# Designing and Standardizing HTTP Response Codes



- 100-199 are informational codes used as low-level signaling mechanisms, such as a confirmation of a request to change protocols
  - 200-299 are general success codes used to describe various kinds of success conditions
  - 300-399 are redirection codes used to request that the consumer retry a request to a different resource identifier, or via a different intermediary
  - 400-499 represent consumer-side error codes that indicate that the consumer has produced a request that is invalid for some reason, example 404 file not found
  - 500-599 represent service-side error codes that indicate that the consumer's request may have been valid but that the service has been unable to process it
- Elasticsearch exposes REST APIs that are used by the UI components and can be called directly to configure and access Elasticsearch features.

Source:

*Service-Oriented Architecture: Analysis and Design for Services and Microservices*, Thomas Erl, 2017

# Idempotence



Idempotence (UK: / d m po tɛns/, [1] US: / a dɛm-/)[2] is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application. The concept of idempotence arises in a number of places in abstract algebra (in particular, in the theory of projectors and closure operators) and functional programming (in which it is connected to the property of referential transparency).

The term was introduced by Benjamin Peirce[3] in the context of elements of algebras that remain invariant when raised to a positive integer power, and literally means "(the quality of having) the same power", from idem + potence (same + power).

- The term idempotent is used multiple times regarding REST and HTTP methods.

Source:

<https://en.wikipedia.org/wiki/Idempotence>

# Computer science examples



A function looking up a customer's name and address in a database is typically idempotent, since this will not cause the database to change. Similarly, changing a customer's address to XYZ is typically idempotent, because the final address will be the same no matter how many times XYZ is submitted. ...

In the Hypertext Transfer Protocol (HTTP), idempotence and safety are the major attributes that separate HTTP verbs. Of the major HTTP verbs, GET, PUT, and DELETE should be implemented in an idempotent manner according to the standard, but POST need not be.[15] GET retrieves a resource; PUT stores content at a resource; and DELETE eliminates a resource. ...

In service-oriented architecture (SOA), a multiple-step orchestration process composed entirely of idempotent steps can be replayed without side-effects if any part of that process fails.

Source:

<https://en.wikipedia.org/wiki/Idempotence>

# ActiveMQ implements a RESTful API



ActiveMQ implements a RESTful API to messaging which allows any web capable device to publish or consume messages using a regular HTTP POST or GET.

To publish a message use a HTTP POST. To consume a message use HTTP DELETE or GET.

You can map a URI to the servlet and then use the relative part of the URI as the topic or queue name. e.g. you could HTTP POST to

```
http://www.acme.com/orders/input
```

which would publish the contents of the HTTP POST to the orders.input queue on JMS.

Similarly you could perform a HTTP DELETE GET on the above URL to read from the same queue. In this case we will map the MessageServlet from ActiveMQ to the URI ...

Note that strict REST requires that GET be a read only operation; so strictly speaking we should not use GET to allow folks to consume messages. Though we allow this as it simplifies HTTP/DHTML/Ajax integration somewhat.

Source: <https://activemq.apache.org/rest>

# Camel book chapter 10: RESTful web services



RESTful web services have become a ubiquitous protocol in recent years and are the topic of chapter 10

We will now move to the Camel book and discuss the concepts presented, and try to not get caught up in all the details

Notes:

- Libraries and programs are often updated
- Some technologies will already be in place when you start working, selected beforehand

Source:

*Camel in action*, Claus Ibsen and Jonathan Anstey, 2018

## 10.1 RESTful services



**Table 10.1** The RESTful API for the Rider Auto Parts order web service

Verb	<code>http://rider.com/orders</code>	<code>http://rider.com/orders/{id}</code>
GET	Retrieves a list of all the orders	Retrieves the order with the given ID
PUT	N/A	Updates the order with the given ID
POST	Creates a new order	N/A
DELETE	Cancels all the orders	Cancels the order with the given ID

RESTful services, also known as REST services, has become a popular architectural style used in modern enterprise projects. REST was defined by Roy Fielding in 2000 when he published his paper, and today REST is a foundation that drives the modern APIs on the web. You can also think of it as a modern web service, in which the APIs are RESTful and HTTP based so they're easily consumable on the web.

Source:

*Camel in action*, Claus Ibsen and Jonathan Anstey, 2018

# Python and REST



```
#!/usr/bin/env python  
import requests  
r = requests.get('https://api.github.com/events')  
print (r.json());
```

- Lets try to use some Python to access a REST service.
- We will use the JSONPlaceholder which is a free online REST API: <https://jsonplaceholder.typicode.com/>
- Start at the site: <https://jsonplaceholder.typicode.com/guide.html> and try running a few of the examples with your browser
- Then try using the same URLs in the Requests HTTP library from Python, <https://requests.readthedocs.io/en/master/>

See how things you can do with the browser, checked manually, can then be automated into programs





We will now dive more into Elasticsearch/elastic stack - being a very useful tool and skill

Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elasticsearch is developed in Java.

Source: <https://en.wikipedia.org/wiki/Elasticsearch>

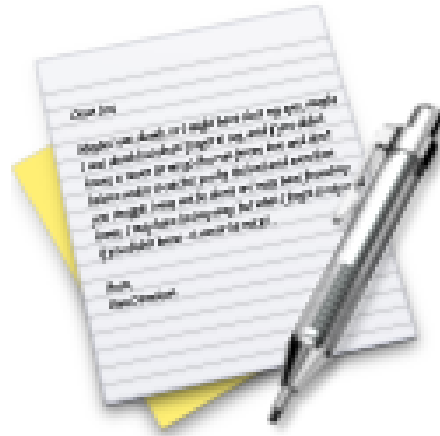
- Open core means parts of the software are licensed under various open-source licenses (mostly the Apache License)
- Various browser tools and plugins for ES exist, to make life easier
- I often use ES for storing Log Messages and Events from multiple systems, a SIEM Security information and event management.

# Elasticsearch REST



- Elasticsearch exposes REST APIs that are used by the UI components and can be called directly to configure and access Elasticsearch features.
- <https://www.elastic.co/guide/en/elasticsearch/reference/current/rest-apis.html>
- So REST is used for putting data, using PUT and POST
- And REST is used for getting data with GET, but also getting information about the Elasticsearch system itself, cluster health etc.
- It supports advanced querying through the API and parallel execution of searches across a cluster of nodes

# Exercise

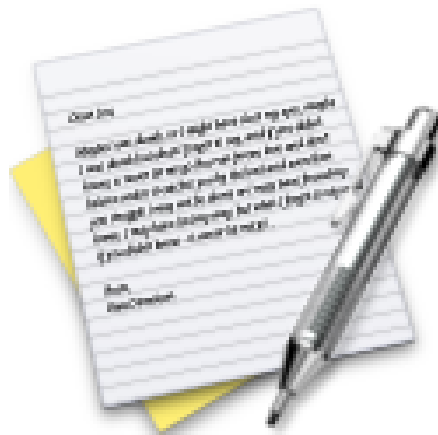


Now lets do the exercise

## Postman API Client 20 min

which is number **15** in the exercise PDF.

# Exercise



Now lets do the exercise

## Install the Elastic Stack - 60 min

which is number **16** in the exercise PDF.

# Exercise

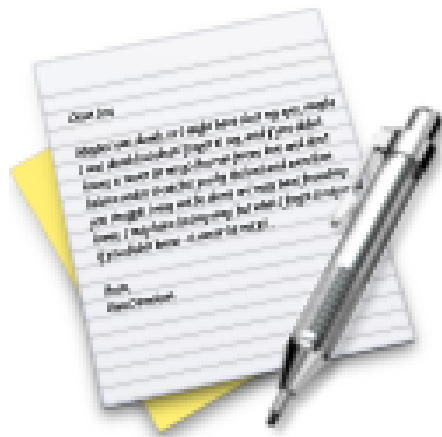


Now lets do the exercise

## Making requests to Elasticsearch - 15-75min

which is number **17** in the exercise PDF.

# Exercise



Now lets do the exercise

## Find Indices in Elasticsearch 15 min

which is number **18** in the exercise PDF.

# Exercise

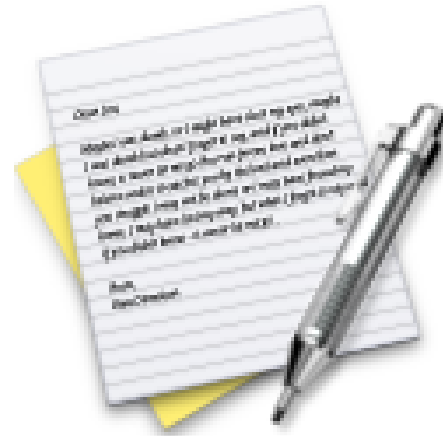


Now lets do the exercise

## Finding Data in Elasticsearch 30 min

which is number **19** in the exercise PDF.

# Exercise



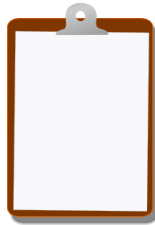
Now lets do the exercise

## Working with dashboards - 30 min

which is number **20** in the exercise PDF.



## For Next Time



Think about the subjects from this time, write down questions

Check the plan for chapters to read in the books

Visit web sites and download papers if needed

Retry the exercises to get more confident using the tools