Welcome to

7. Modern patterns and services: Microservices and REST

KEA System Integration F2020 10 ECTS

Henrik Kramselund Jereminsen hkj@zencurity.com @kramse

Slides are available as PDF, kramse@Github

7-Modern-patterns-system-integration.tex in the repo security-courses

This weeks Agenda in system integration

- Follow the plan:
  `https://zencurity.gitbook.io/kea-it-sikkerhed/system-integration/lektionsplan`
- Work on the hand-in assignment I: Describe the system environment for an organisation
- Plan for April 20.
  I will start with repeating some of the key points from the previous days and exercises. Then I will go through the planned subjects according to the plan
- Reading, as announced since beginning of April is still SOA chapter 6-7, Camel chapter 7,10 which will be the subjects for today

Goals for today

Todays goals:

- Get back into System Integration
- Repeat some of the slides and exercises
- See how applications can run more independently as microservices and REST

Photo by Thomas Galler on Unsplash

April 20. Time schedule

- 08:30 2x 45 min with 10min break
  Repeat some of the previous parts, including the exercises I asked you to perform from the exercises booklet: 10. Run PostgreSQL, 11. Why go to SOA, 12. Cloud Computing Introduction, 13. Cloud Deployment 14. Download the Microservices ebook
- 10:15 2x 45 min with 10min break
  Subject Microservices, going over the parts from late march SOA chapter 6, Camel chapter 7, Microservices for Java chapter 1 We will be working through some of the examples from the Camel book chapter 7 using your Debian with JDK 8. These are running examples of "microservices".
- 12:30 2x 45min with 10min break
  Subject REST SOA chapter 7, Camel chapter 10 We will also be doing small exercises with Python and rest
- 14:15 Optional 45 min
  Chatting, doing exercises, questions about Linux, Camel whatever. Loose and optional, and talking about the Hand-in assignment I: Describe the system environ- ment for an organisation.

  So now going back to 6-1-SOABOOK-system-integration.pdf

Plan for this slide show

- Modern patterns and services
- Microservices
- REST

Exercises

- Mostly talking about the concepts, the Camelbook apps and services
  - since running the examples may prove difficult, and the concepts are more important than getting a small example running.

Reading Summary

Microservices for Java chapter 1

SOA chapter 6: Analysis and Modeling with Web Services and Microservices

Camel book chapter 7: Microservices

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

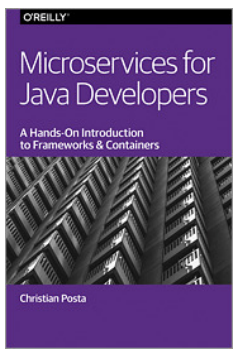
Camel book chapter 10:RESTfull web services

Camel in action, Claus Ibsen and Jonathan Anstey, 2018 ISBN: 978-1-61729-293-4

Service-Oriented Architecture: Analysis and Design for Services and Microservices, Thomas Erl, 2017 ISBN: 978-0-13-385858-7

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, and the Java book as a real-life example from people that have done this in production

Microservices for Java chapter 1



Microservices for Java Developers , Christian Posta, 2016 O'Reilly
`https://www.oreilly.com/programming/free/files/microservices-for-java-developers.pdf`

We will use the introduction, which is recommended.

Whats in the book

> This book is for Java developers and architects interested in developing microservices. We start the book with the high-level understanding and fundamental prerequisites that should be in place to be successful with a microservice architecture.

Introducing some Java frameworks

- The Spring ecosystem, Dropwizard and WildFly Swarm, we'll use JBoss Forge CLI
- Finally, when we build and deploy our microservices as Docker containers running inside of Kubernetes
- They use VirtualBox with Docker and Kubernetes, YMMV
- With source on Github, not updated in 4 years though!
  `https://github.com/redhat-developer/microservices-by-example-source`

Source:

Microservices for Java Developers , Christian Posta, 2016 O'Reilly

Open source is also leading the charge in the technology space

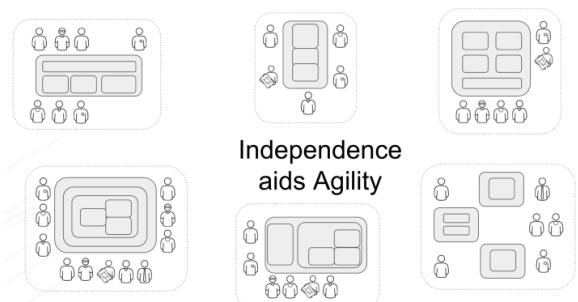
> Open source is also leading the charge in the technology space. Following the commoditization curves, open source is a place developers can go to challenge proprietary vendors by building and innovating on software that was once only available (without source no less) with high license costs. This drives communities to build things like operating systems (Linux), programming languages (Go), message queues (Apache ActiveMQ), and web servers (httpd). Even companies that originally rejected open source are starting to come around by open sourcing their technologies and contributing to existing communities. As open source and open ecosystems have become the norm, we're starting to see a lot of the innovation in software technology coming directly from open source communities (e.g., Apache Spark, Docker, and Kubernetes).

- We have used Linux and multiple products from the Apache website/foundation

Source:

Microservices for Java Developers , Christian Posta, 2016 O'Reilly

Building distributed systems is hard



Microservice architecture (MSA) is an approach to building software systems that decomposes business domain models into smaller, consistent, bounded-contexts implemented by services. These services are isolated and autonomous yet communicate to provide some piece of business functionality. Microservices are typically implemented and operated by small teams with enough autonomy that each team and service can change its internal implementation details (including replacing it outright!) with minimal impact across the rest of the system.

Source:

Microservices for Java Developers , Christian Posta, 2016 O'Reilly

Teamwork

- Teams communicate through promises
- specify these promises with interfaces of their services and via wikis that document their services
- Each team would be responsible for designing the service, picking the right technology for the problem set, and deploying, managing and waking up at 2 a.m. for any issues
- Understand what the service is doing without being tangled into other concerns in a larger application
- Quickly build the service locally
- Pick the right technology for the problem (lots of writes? lots of queries? low latency? bursty?)
- Test the service
- Build/deploy/release at a cadence necessary for the business, which may be independent of other services
- Identify and horizontally scale parts of the architecture where needed
- Improve resiliency of the system as a whole

Source:

Microservices for Java Developers , Christian Posta, 2016 O'Reilly

Challenges

- Microservices may not be efficient. It can be more resource intensive.
- You may end up with what looks like duplication.
- Operational complexity is a lot higher.
- It becomes very difficult to understand the system holistically.
- It becomes significantly harder to debug problems.
- In some areas you may have to relax the notion of transaction.

Source:

Microservices for Java Developers , Christian Posta, 2016 O'Reilly

Design for Faults

> Things will fail, so we must develop our applications to be resilient and handle failure, not just prevent it. We should be able to deal with faults gracefully and not let faults propagate to total failure of the system.
>
> Building distributed systems is different from building shared- memory, single process, monolithic applications. One glaring differ- ence is that communication over a network is not the same as a local call with shared memory.

- Networks are inherently unreliable

Source:

Microservices for Java Developers , Christian Posta, 2016 O'Reilly

Design with Dependencies in Mind

> To be able to move fast and be agile from an organization or distributed-systems standpoint, we have to design systems with dependency thinking in mind; we need loose coupling in our teams, in our technology, and our governance.

Source:

Microservices for Java Developers , Christian Posta, 2016 O'Reilly

Design with Promises in Mind

> In a microservice environment with autonomous teams and serv- ices, it's very important to keep in mind the relationship between service provider and service consumer. As an autonomous service team, you cannot place obligations on other teams and services because you do not own them; they're autonomous by definition. All you can do is choose whether or not to accept their promises of functionality or behavior. As a provider of a service to others, all you can do is promise them a certain behavior.

- Promises as published by APIs and versions in those!
- References *Consumer-Driven Contracts: A Service Evolution Pattern*
  `https://martinfowler.com/articles/consumerDrivenContracts.html`

Source:

Microservices for Java Developers , Christian Posta, 2016 O'Reilly

SOA chapter 6:

Analysis and Modeling with Web Services and Microservices

> This chapter provides a detailed step-by-step process for modeling Web service candidates.
> 6.1 Web Service Modeling Process

- Chapter goes through the steps of a service modelling process
- End result is utility service, microservices and non-agnostic services - process specific logic

Source:

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

Step 9: Define Microservice Candidates

As discussed in Chapter 4, the microservice model can introduce a highly independent and autonomous service implementation architecture that can be suitable for units of logic with particular processing demands.

Typical considerations can include:

- Increased autonomy requirements
- Specific runtime performance requirements
- Specific runtime reliability or failover requirements
- Specific service versioning and deployment requirements

Source:

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

The revised service composition candidate

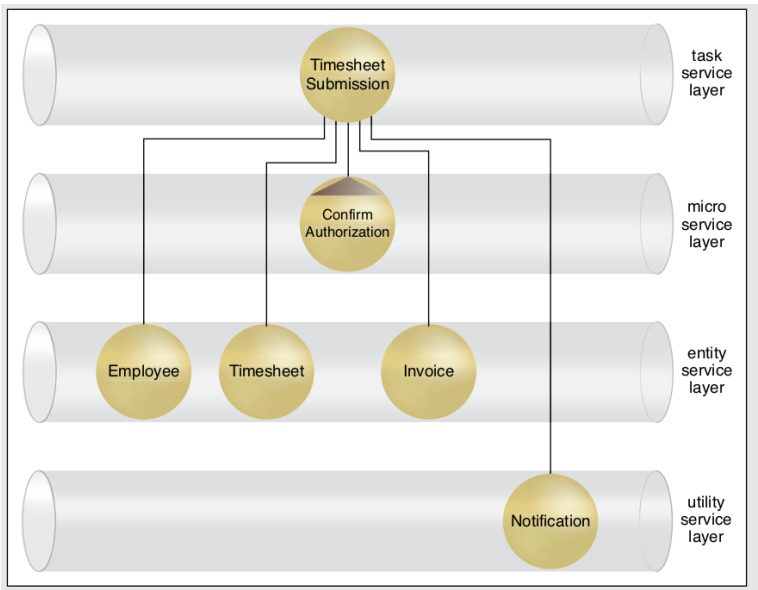


Figure 6.12
The revised service composition candidate incorporating the new utility service and microservice.

Source:

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

SOA book chapter 7:

Analysis and Modeling with REST Services and Microservices

7.1 REST Service Modeling Process
7.2 Additional Considerations

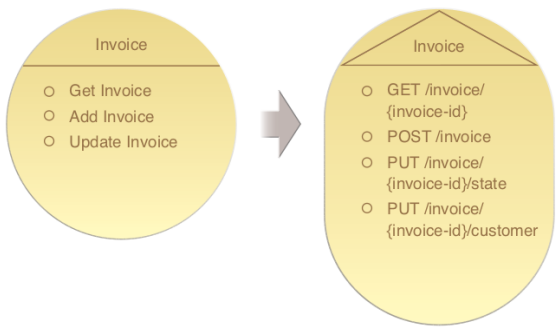
- This chapter provides a detailed step-by-step process for modeling REST service candidates.
- End result is similar to chapter 6 a service composition candidate

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`

Source:

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

Camel book chapter 7: Microservices

Camel is ideal for building microservice applications, which is the topic of chapter 7. The chapter has many examples that demonstrate how to use Camel with popular microservice runtimes such as Spring Boot and WildFly Swarm.

Source:

Camel in action, Claus Ibsen and Jonathan Anstey, 2018

7.2 Running Camel microservices

Which microservice runtimes does Camel support? The answer is all of them. Camel is just a library you include in the JVM runtime, and it runs anywhere. This section walks you through running Camel in some of the most popular microservice runtimes:

- Standalone Running just Camel
- CDI-Running Camel with CDI
- WildFly Swarm-We's see how Camel runs with the lightweight Java EE server
- Spring Boot-Running Camel with Spring Boot

Source:

Camel in action, Claus Ibsen and Jonathan Anstey, 2018

Camel book chapter 10: RESTfull web services

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

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	http://rider.com/orders	http://rider.com/orders/{id}
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

Moving to the Camel book

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

Note:

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

Source:

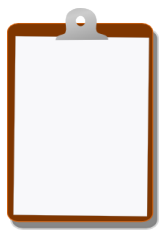
Camel in action, Claus Ibsen and Jonathan Anstey, 2018

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/`

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