Welcome to

11. Integration examples and standards

KEA System Integration F2020 10 ECTS

Henrik Kramselund Jereminsen hkj@zencurity.com @kramse

Slides are available as PDF, kramse@Github

11-integration-examples-system-integration.tex in the repo security-courses

Goals for today

Todays goals:

- Finish Enterprise Integration Patterns book
- Talk about securing, running and deploying Camel
- Discuss how to manage and monitor production environment

Photo by Thomas Galler on Unsplash

Time schedule

- 08:15 - 11:30
  EIP Chapter 13: Case Study: Bond Trading System Chapter 14: Concluding Remarks
  Finish Enterprise Integration Patterns
- Camel chapter 14: Securing Camel
  Chapter 15: Running and deploying Camel
- 12:15 -13:45 After lunch
  Camel chapter 16: Management and Monitoring
- Exercises in course so far. Repeat Ansible, how to setup production systems

Plan for today

- Integration examples and standards
- Running Camel integration Management, Logs, Monitoring, Security
- - using systems we have used in this course as examples

Exercises

- Exercises in course so far.
- Repeat Ansible
- How to setup production systems

Reading Summary

EIP 13-14

Camel ch 14-16

SOA Appendix A

EIP chapter 13: Integration Patterns in Practice

This chapter covers

- Case Study: Bond Trading System
- Architecture with Patterns
- Problem Solving With Patterns

Source:

Enterprise Integration Patterns, Gregor Hohpe and Bobby Woolf, 2004

ISBN: 978-0-321-20068-6

Note: Chapter content is available at:
`https://www.enterpriseintegrationpatterns.com/patterns/messaging/BondTradingCaseStudy.html`

EIP chapter 14: Concluding Remarks

This chapter covers

- Finishing up the book
- List various standard groups

Source:

Enterprise Integration Patterns, Gregor Hohpe and Bobby Woolf, 2004

ISBN: 978-0-321-20068-6

Note: Chapter content is available at:
`https://www.enterpriseintegrationpatterns.com/patterns/messaging/Future.html`

Camel chapter 14: Securing Camel

This chapter covers

- Securing your Camel configuration
- Web service security
- Transport security
- Encryption and decryption
- Signing messages
- Authentication and authorization

Source:

Camel in action, Claus Ibsen and Jonathan Anstey, 2018, 2nd edition ISBN: 978-1-61729-293-4

Camel chapter 15: Running and deploying Camel

This chapter covers

- Starting and stopping Camel safely
- Adding and removing routes at runtime
- Deploying Camel
- Running standalone
- Running in web containers
- Running in Java EE servers
- Running with OSGi
- Running with CDI

Source:

Camel in action, Claus Ibsen and Jonathan Anstey, 2018, 2nd edition ISBN: 978-1-61729-293-4

Camel chapter 16: Management and Monitoring

This chapter covers

- Monitoring Camel instances
- Tracking application activities
- Using notifications
- Managing Camel applications with JMX and REST
- Understanding and using the Camel management API
- Gathering runtime performance statistics
- Using Dropwizard metrics with Camel
- Developing custom components for management

Source:

Camel in action, Claus Ibsen and Jonathan Anstey, 2018, 2nd edition ISBN: 978-1-61729-293-4

SOA Appendix A: Service-Orientation Principles Reference

This appendix provides profile tables for the patterns referenced throughout this book. As explained in Chapter 1, each pattern reference is suffixed with the page number of its corresponding profile table in this appendix.

Similar to Appendix B: SOA Design Patterns Reference

Source:

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

Thomas Erl, 2017 ISBN: 978-0-13-385858-7

Design patterns are helpful

Design patterns are helpful because they:

- Represent field-tested solutions to common design problems
- Organize design intelligence into a standardized and easily “referenceable” format
- Are generally repeatable by most IT professionals involved with design
- Can be used to ensure consistency in how systems are designed and built
- Can become the basis for design standards
- Are usually fl exible and optional (and openly document the impacts of their appli- cation and even suggest alternative approaches)
- Can be used as educational aids by documenting specific aspects of system design (regardless of whether they are applied)
- Can sometimes be applied prior and subsequent to the implementation of a system
- Can be supported via the application of other design patterns that are part of the same collection
- Enrich the vocabulary of a given IT field because each pattern is given a meaningful name

Part I 08:30 2x 45 min

EIP Chapter 13 and 14 Finish Enterprise Integration Patterns

Case Study: Bond Trading System

This chapter covers

- Case Study: Bond Trading System
- Architecture with Patterns
- Problem Solving With Patterns

Source:

Enterprise Integration Patterns, Gregor Hohpe and Bobby Woolf, 2004

ISBN: 978-0-321-20068-6

Note: We will now continue at the book site:

`https://www.enterpriseintegrationpatterns.com/patterns/messaging/BondTradingCaseStudy.html`

Concluding Remarks

- Emerging Standards and Futures in Enterprise Integration

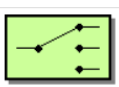
  Note: We will now continue at the book site:

  `https://www.enterpriseintegrationpatterns.com/patterns/messaging/Future.html`

Further patterns

- Further patterns, newer data and articles are available from the authors:
  `https://www.enterpriseintegrationpatterns.com/ramblings.html`

Modern Examples for Enterprise Integration Patterns

	Publish-Subscribe Channel	Google Cloud Pub/sub
	Dead Letter Channel	Amazon SQS
	Return Address	GoLang
	Content-based Router	Apache Camel
	Message Filter	RabbitMQ
	Event-driven Consumer	RabbitMQ
	Competing Consumers	Apache Kafka
	Channel Purger	Amazon SQS

Modern Examples for Enterprise Integration Patterns

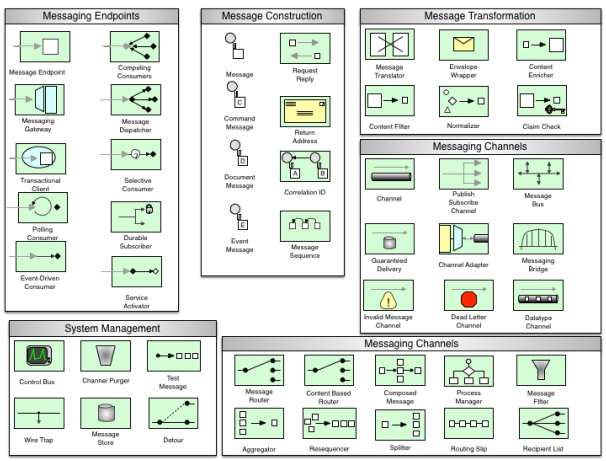
`https://www.enterpriseintegrationpatterns.com/ramblings/eip1_examples_updated.html`

What Products Implement or Use Enterprise Integration Patterns?

The patterns are not tied to a specific implementation. They help you design better solutions, whether you use any of the following platforms:

- **EAI and SOA platforms**, such as IBM WebSphere MQ, TIBCO, Vitria, Oracle Service Bus, WebMethods (now Software AG), Microsoft BizTalk, or Fiorano.
- **Open source ESB's** like Mule ESB, JBoss Fuse, Open ESB, WSo2, Spring Integration, or Talend ESB
- **Message Brokers** like ActiveMQ, Apache Kafka, or RabbitMQ
- **Web service- or REST-based integration**, including Amazon Simple Queue Service (SQS) or Google Cloud Pub/Sub
- **JMS-based messaging systems**
- **Microsoft technologies** like MSMQ or Windows Communication Foundation (WCF)

Stencils for EIP



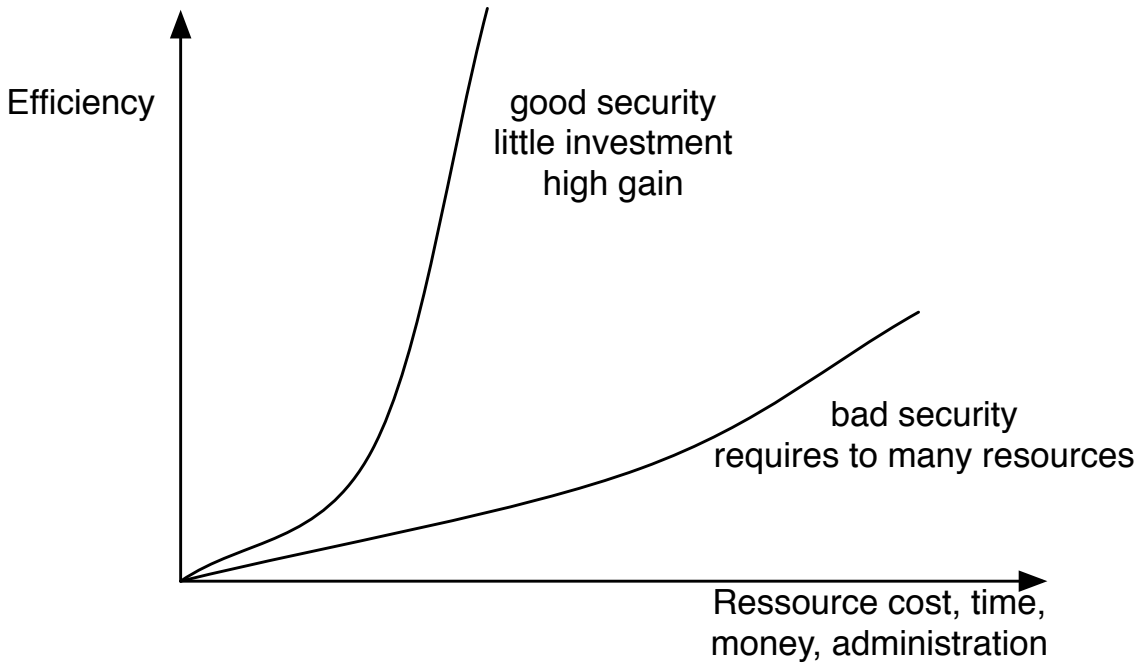
Document. You can create design documents using our icon language by downloading the Visio stencil or using the OmniGraffle stencil created by one of our readers.

- `https://www.enterpriseintegrationpatterns.com/patterns/messaging/downloads.html`
- `http://www.graffletopia.com/stencils/137`

Part II 10:15 2x 45 min

Camel chapter 14-15

Good security



You always have limited resources for protection - use them as best as possible

Recommendations

Keep updated!

- read web sites, books, articles, mailing lists, Twitter, ...

Always have a chapter on security evaluation

- any process must have security, like RFC Request for Comments have

Incident Response

- you WILL have security incidents, be prepared

Write down security policy

- including software and e-mail policies

Advice

Use technology

Learn the technology - read the freaking manual

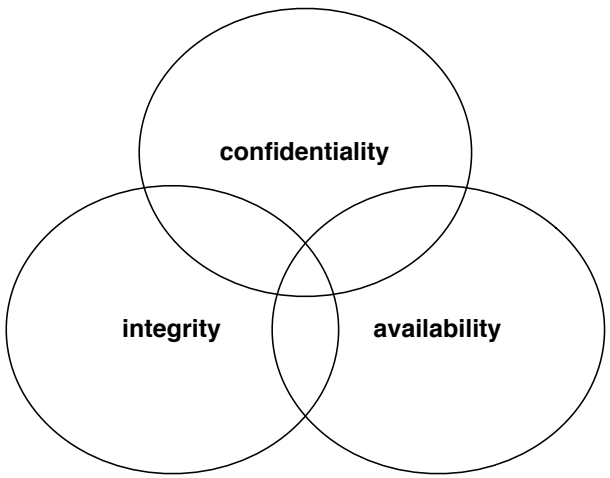
Think about the data you have, upload, facebook license?! WTF!

Think about the data you create - nude pictures taken, where will they show up?

- Turn off features you don't use
- Turn off network connections when not in use
- Update software and applications
- Turn on encryption: IMAP**S**, POP3**S**, HTTP**S** also for data at rest, full disk encryption, tablet encryption
- Lock devices automatically when not used for 10 minutes
- Dont trust fancy logins like fingerprint scanner or face recognition on cheap devices

But which features to disable? Let the security principles guide you

Confidentiality, Integrity and Availability



We want to protect something

Confidentiality - data kept a secret

Integrity - data is not subjected to unauthorized changes

Availability - data and systems are available when needed

Security is a process

Remember:

- what is information and security?
- Data kept electronically
- Data kept in physical form
- Dont forget the human element of security

Incident Response and Computer Forensics reaction to incidents

Good security is the result of planning and long-term work

Security is a process, not a product, Bruce Schneier

Source for quote: `https://www.schneier.com/essays/archives/2000/04/the_process_of_secur.html`

Work together

Team up!

We need to share security information freely

We often face the same threats, so we can work on solving these together

Goals of Security

Prevention - means that an attack will fail

Detection - determine if attack is underway, or has occured - report it

Recovery - stop attack, assess damage, repair damage

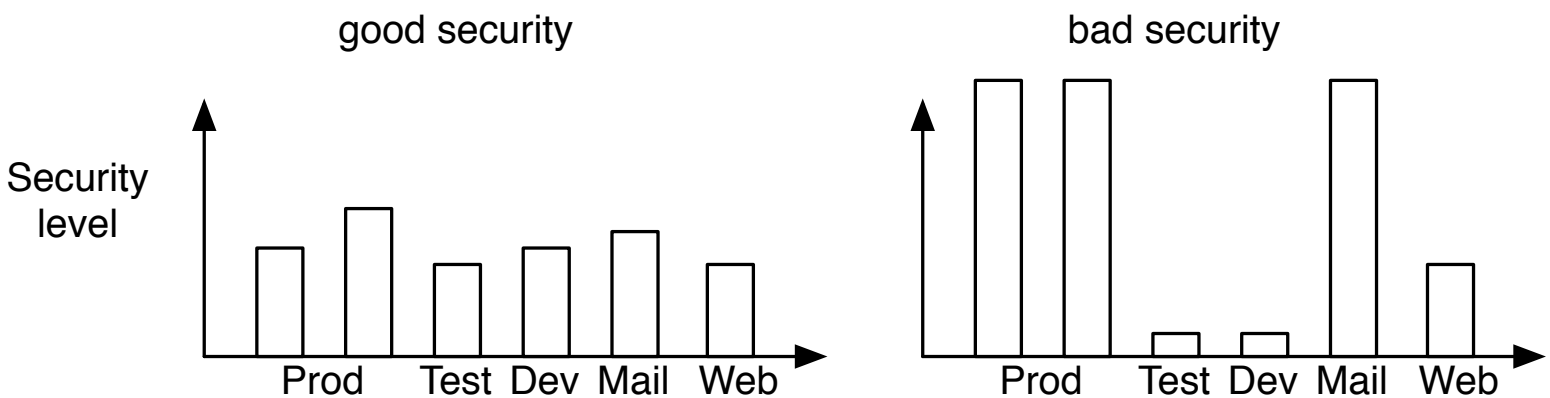
Policy and Mechanism

Definition 1-1. A *security policy* is a statement of what is, and what is not, allowed.

Definition 1-2. A *security mechanism* is a method, tool or procedure for enforcing a security policy.

Quote from Matt Bishop, Computer Security section 1.3

Balanced security



Better to have the same level of security

If you have bad security in some part - guess where attackers will end up

Hackers are not required to take the hardest path into the network

Realize there is no such thing as 100% security

Cost-Benefit Analysis

Benefits of computer security must be weighed against value of assets

Often more expensive to add security mechanisms to a system, than designing them in

Risk management defined

Information Risk Management

Life is full of risk.

Risk is the possibility of damage happening and the ramifications of such damage should it occur. ***Information risk management (IRM)*** is the *process* of identifying and assessing risk, reducing it to an acceptable level, and implementing the right mechanisms to maintain that level. There is no such thing as a 100 percent secure environment. Every environment has vulnerabilities and threats to a certain degree. The skill is in identifying these threats, assessing the probability of them actually occurring and the damage they could cause, and then taking the right steps to reduce the overall level of risk in the environment to what the organization identifies as acceptable.

Source: Shon Harris *CISSP All-in-One Exam Guide*

Securing your Camel configuration

```
[janstey@bender]$ cd apache-camel-2.20.1/
[janstey@bender]$ wget http://repo1.maven.org/maven2/org/jasypt/jasypt/1.9.2/
jasypt-1.9.2.jar
[janstey@bender]$ java -cp jasypt-1.9.2.jar:lib/camel-jasypt-2.20.1.jar org.
apache.camel.component.jasypt.Main -help
Apache Camel Jasypt takes the following options

-help = Displays the help screen
-command <command> = Command either encrypt or decrypt
-password <password> = Password to use
-input <input> = Text to encrypt or decrypt
-algorithm <algorithm> = Optional algorithm to use
```

- Encrypting configuration is possible using `camel-jasypt`
- Maybe not recommended way to do this now

Web service security

Web services are an extremely useful integration technology for distributed applica- tions. They are often associated with service-oriented architecture (SOA), in which each service is defined as a web service.

- TL;DR Almost everything is a web service today, even Mobile ppps
- Chapter is really inadequate for describing real security
- Never use unencrypted FTP and HTTP, unfortunately it happens ...
- Recommend The Open Web Application Security Project (OWASP) instead `https://owasp.org/`

Transport security

- Transport Security is needed for HTTP, making it HTTPS
- In Java, TLS is typically configured using the Java Secure Socket Exten- sion (JSSE) API provided with the JRE

Cryptography

Cryptography or cryptology is the practice and study of techniques for secure communication

Modern cryptography is heavily based on mathematical theory and computer science practice; cryptographic algorithms are designed around computational hardne ss assumptions, making such algorithms hard to break in practice by any adversary

Symmetric-key cryptography refers to encryption methods in which both the sender and receiver share the same key, to ensure confidentiality, example algorit hm AES

Public-key cryptography (like RSA) uses two related keys, a key pair of a public key and a private key. This allows for easier key exchanges, and can provid e confidentiality, and methods for signatures and other services

Source: `https://en.wikipedia.org/wiki/Cryptography`

DES, Triple DES og AES

AES

Advanced Encryption Standard

DES - old and retired!!

In 2001 a newer standard was adopted Advanced Encryption Standard (AES)

It replaces Data Encryption Standard (DES)

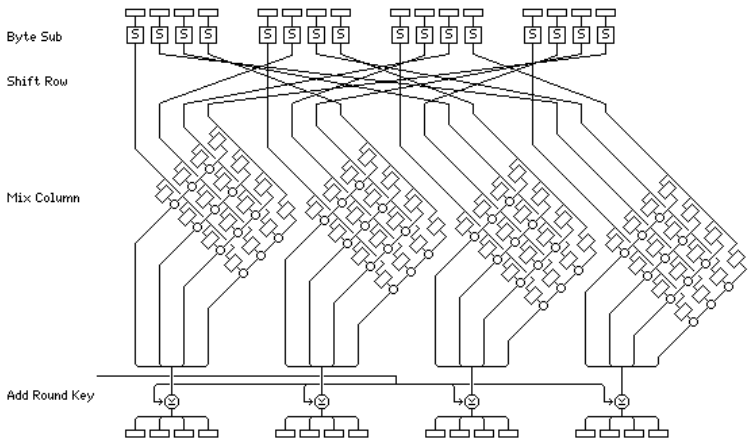
Algorithm is Rijndael developed by Joan Daemen og Vincent Rijmen.

See `https://en.wikipedia.org/wiki/Advanced_Encryption_Standard`

Animations can be found (including errors)

`https://www.youtube.com/watch?v=mlzxpkdXP58`

AES Advanced Encryption Standard



- The official Rijndael web site displays this image to promote understanding of the Rijndael round transformation [8].
- Key sizes 128,192,256 bit typical
- Some extensions in cryptosystems exist: XTS-AES-256 really is 2 instances of AES-128 and 384 is two instances of AES-192 and 512 is two instances of AES-256
- `https://en.wikipedia.org/wiki/RSA_(cryptosystem)`

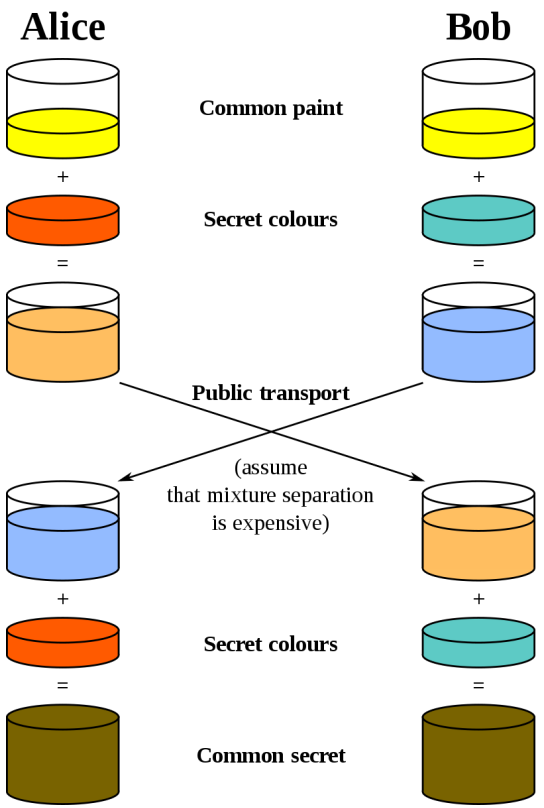
RSA

> RSA (Rivest–Shamir–Adleman) is one of the first public-key cryptosystems and is widely used for secure data transmission. ... In RSA, this asymmetry is based on the practical difficulty of the factorization of the product of two large prime numbers, the "factoring problem". The acronym RSA is made of the initial letters of the surnames of Ron Rivest, Adi Shamir, and Leonard Adleman, who first publicly described the algorithm in 1978.

- Key sizes 1,024 to 4,096 bit typical
- Quote from: `https://en.wikipedia.org/wiki/RSA_(cryptosystem)`

Diffie Hellman exchange

Diffie–Hellman key exchange (DH)[nb 1] is a method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as originally conceptualized by Ralph Merkle and named after Whitfield Diffie and Martin Hellman.[1][2] DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography. ... The scheme was first published by Whitfield Diffie and Martin Hellman in 1976



- Quote from: `https://en.wikipedia.org/wiki/Diffie-Hellman_key_exchange`
- Today we also use elliptic curves with DH
  `https://en.wikipedia.org/wiki/Elliptic-curve_cryptography`

Elliptic Curve

> Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-EC cryptography (based on plain Galois fields) to provide equivalent security.[1]

- Today we use `https://en.wikipedia.org/wiki/Elliptic-curve_cryptography`

Transport Layer Security (TLS)

Oprindeligt udviklet af Netscape Communications Inc.

Secure Sockets Layer SSL er idag blevet adopteret af IETF og kaldes derfor også for Transport Layer Security TLS TLS er baseret på SSL Version 3.0

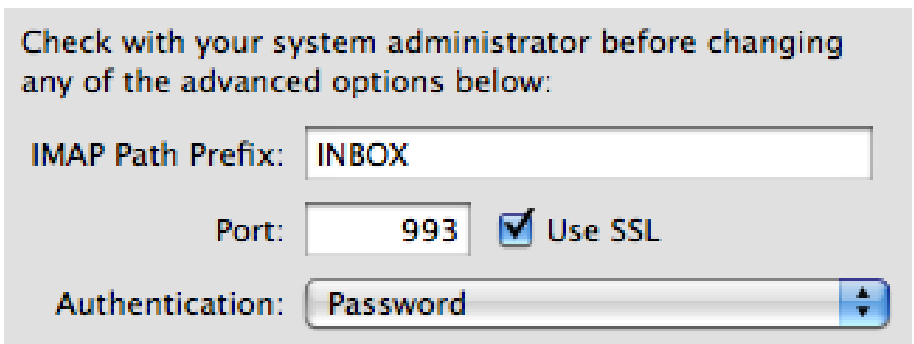
RFC-2246 The TLS Protocol Version 1.0 fra Januar 1999

RFC-3207 SMTP STARTTLS

Det er svært!

Stanford Dan Boneh udgiver en masse omkring crypto
`https://crypto.stanford.edu/~dabo/cryptobook/`

SSL/TLS protocols



- Many protocols have been extended to cover TLS
- HTTPS vs HTTP
- IMAPS, POP3S, osv.
- Some use the same port, others use two different ports IMAP 143/tcp vs IMAPS 993/tcp
- Those using the same port often can use START TLS, like:
  SMTP STARTTLS RFC-3207

Encryption and Transport Security Summary

> Camel is unsecured by default—Probably the most important point is that Camel has no security settings turned on by default. This is great for development, but before your application is deployed in the real world, you'll most likely need some form of security enabled.

- You need to use Transport Layer Security (TLS)
- I recommend using Let's Encrypt `https://letsencrypt.org/`

Source:

Camel in action, Claus Ibsen and Jonathan Anstey, 2018, 2nd edition ISBN: 978-1-61729-293-4

Chapter 15: Running and deploying Camel

This chapter covers

- Starting and stopping Camel safely
- Adding and removing routes at runtime
- Deploying Camel
- Running standalone
- Running in web containers
- Running in Java EE servers
- Running with OSGi
- Running with CDI

Deploying Camel

This section presents five deployment strategies that are possible with Camel and their strengths and weaknesses:

- Embedding Camel in a Java application
- Running Camel in a web environment such as Apache Tomcat
- Running Camel inside WildFly
- Running Camel in an OSGi container such as Apache Karaf
- Running Camel in a container that supports CDI, such as Apache Karaf or WildFly

Source:

Camel in action, Claus Ibsen and Jonathan Anstey, 2018, 2nd edition ISBN: 978-1-61729-293-4

Deploying to Apache Tomcat

```
[janstey@ghost apache-tomcat-8.5.23]$ bin/startup.sh
Using CATALINA_BASE: /home/janstey/kits/apache-tomcat-8.5.23
Using CATALINA_HOME: /home/janstey/kits/apache-tomcat-8.5.23
Using CATALINA_TMPDIR: /home/janstey/kits/apache-tomcat-8.5.23/temp
Using JRE_HOME:
/usr/java/jdk1.8.0_91/
Using CLASSPATH:
/home/janstey/kits/apache-tomcat-8.5.23/bin/bootstrap.
jar:/home/janstey/kits/apache-tomcat-8.5.23/bin/tomcat-juli.jar
Tomcat started.
```

- First starting Tomcat, we did this early in the course
- Then copy war file - Java Archive with Web Application
  `cp target/riderautoparts-war-2.0.0.war ~/kits/apache-tomcat-8.5.23/webapps/`

Part III 12:30 2x 45min

Camel chapter 16: Management and monitoring

Monitoring Camel instances

- *Network level* – This is the most basic level, where you check that the network connectivity is working.
- *JVM level* – At this level, you check the JVM that hosts the Camel application. The JVM exposes a standard set of data using the JMX technology.
- *Application level* – Here you check the Camel application using JMX or other techniques.
- JConsole can be used for JMX connection
- Read `https://tomcat.apache.org/tomcat-9.0-doc/monitoring.html` about Java Management Extensions (JMX) `https://en.wikipedia.org/wiki/Java_Management_Extensions`
  Note: references to RMI, JMS, SNMP, HTTP etc.

Logs: Tracking application activities

```
*.err;kern.debug;auth.notice;authpriv.none;mail.crit    /dev/console
*.notice;auth,authpriv,cron,ftp,kern,lpr,mail,user.none /var/log/messages
kern.debug;user.info;syslog.info                        /var/log/messages
auth.info                                               /var/log/authlog
authpriv.debug                                          /var/log/secure
...
# Uncomment to log to a central host named "loghost".
#*.notice;auth,authpriv,cron,ftp,kern,lpr,mail,user.none        @loghost
kern.debug,user.info,syslog.info                                @loghost
auth.info,authpriv.debug,daemon.info                            @loghost
```

Centralized syslogging is recommended!

Kibana

Highly recommended for a lot of data visualisation

Non-programmers can create, save, and share dashboards

Source: `https://www.elastic.co/products/kibana`

Logstash pipeline

Logstash is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite "stash." (Ours is Elasticsearch, naturally.)
`https://www.elastic.co/products/logstash`

```
input { stdin { } }
output {
  elasticsearch { host => localhost }
  stdout { codec => rubydebug }
}
```

Grok expresssions

```
filter {
  if [type] == "syslog" {
    grok {
      match => { "message" => "%{SYSLOGTIMESTAMP:syslog_timestamp}
      %{SYSLOGHOST:syslog_hostname} %{DATA:syslog_program}
      (?:\[%{POSINT:syslog_pid}\])?: %{GREEDYDATA:syslog_message}" }
      add_field => [ "received_at", "%{@timestamp}" ]
      add_field => [ "received_from", "%{host}" ]
    }
    syslog_pri { }
    date {
      match => [ "syslog_timestamp", "MMM  d HH:mm:ss", "MMM dd HH:mm:ss" ]
    }
  }
}
```

- Logstash filter expressions grok can normalize and split data into fields

Source: Config snippet from recommended link

`http://logstash.net/docs/1.4.1/tutorials/getting-started-with-logstash`

Managing Camel applications

- *Camel application lifecycle* – Control your Camel application, such as stopping and starting routes, and much more using a broad range of ways with JMX, Jolokia, hawtio, and the ControlBus component
- *Camel management API* – Learn about the programming API from Camel that defines the management API.
- *Performance statistics* – Discover which key metrics Camel captures about your Camel application performance, and how to access these metrics for custom reporting and hook into monitoring and alert tools.
- *Management enabling custom components* – Learn to program your custom Camel components and Java beans so they're management enabled out of the box, as if they were first-class from the Camel release.

Gathering runtime performance statistics

Book has lots of examples with hawtio and JConsole

You can try this example by running the following Maven goal from the chapter16/ custom-bean directory:

```
mvn compile exec:java
```

If you run the example and connect to the JVM using JConsole, you can find the custom bean in the JMX tree under the Camel processor tree, as shown in figure 16.16.

Production setup

> The Prometheus trait configures the Prometheus JMX exporter and exposes the integration with a Service and a ServiceMonitor resources so that the Prometheus endpoint can be scraped.

- I would propose looking into Prometheus instead
- `https://camel.apache.org/camel-k/latest/traits/prometheus.html`

Modern Monitoring

- Prometheus collects metrics from monitored targets by scraping metrics HTTP endpoints on these targets. Since Prometheus also exposes data in the same manner about itself, it can also scrape and monitor its own health.
  `https://prometheus.io/`
- Grafana is the open source analytics and monitoring solution for every database
  `https://grafana.com/`
- Loki is a horizontally-scalable, highly-available, multi-tenant log aggregation system inspired by Prometheus.
  `https://grafana.com/oss/loki/`

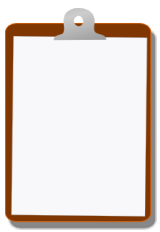
Part IV 12:30 2x 45min

Exercises in course so far. Repeat Ansible, how to setup production systems

Let's run some code, repeat or something new. Not required for the exam, but helps understand the systems:

- Repeat the Ansible setup, re-run Kramse Labs Ansible playbooks
- Repeat Logstash, Elasticsearch and Kibana tasks – creating a dashboard perhaps
- Run Tomcat some more, deploy a war
- Perhaps try `https://prometheus.io/docs/prometheus/latest/getting_started/`

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