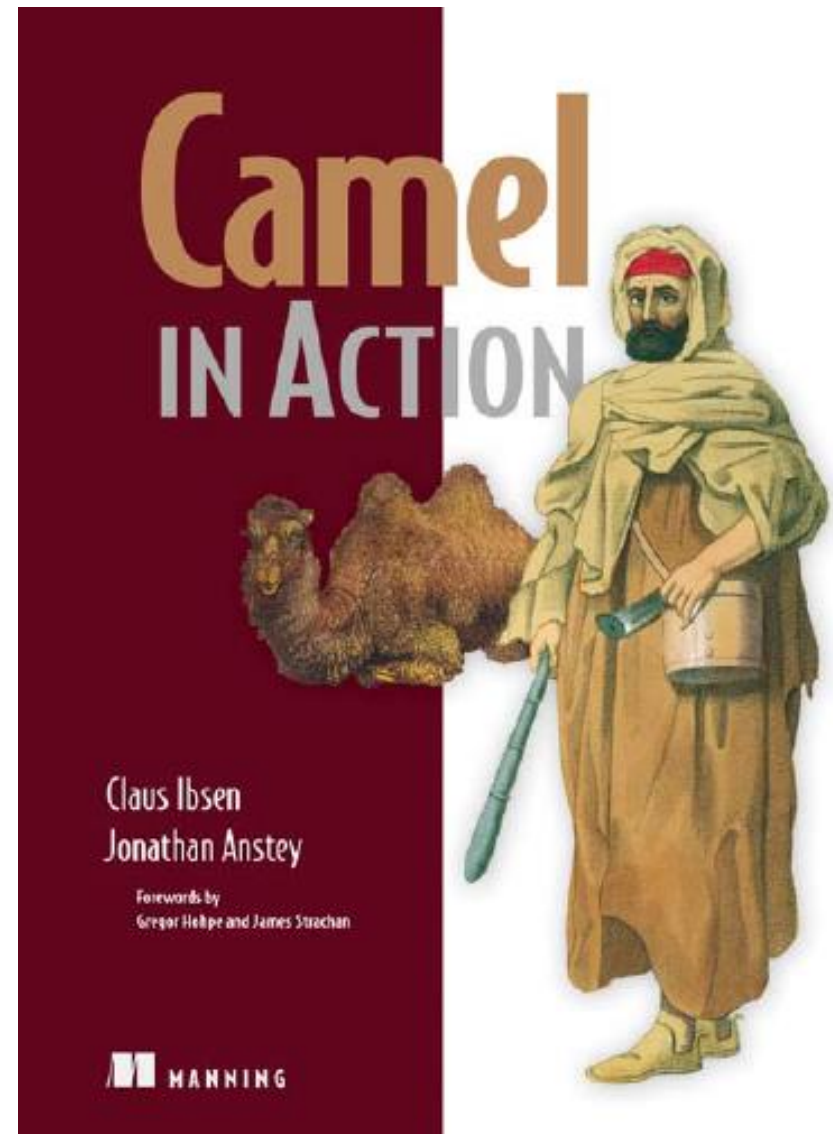


JAVACREAM

*Training
Consulting
Projectmanagement*

Apache Service Mix

Ein Java-basierter Integrationsplattform



- Die in diesem Seminar verwendete Werkzeuge und Frameworks sind Open Source
 - LPGL Lizenzmodell
- Dies ist ein Programmier-Seminar
 - Damit werden die Inhalte durch Übungen vertieft und verinnerlicht
 - Musterbeispiele werden zur Verfügung gestellt
 - Diese können am Ende des Seminars als ZIP-Datei kopiert werden
 - USB-Stick oder ähnliches
- Dokumentation und Ressourcen stehen auch im Internet zur Verfügung
- Konventionen
 - Befehle werden in `Courier-Schriftart` dargestellt
 - Dateinamen werden in *`Courier-Schriftart`* dargestellt
 - Links werden in `Courier-Schriftart` dargestellt
 - Zitate werden in *"Anführungszeichen kursiv"* formatiert, die Quellenangabe steht eingerückt darunter

© Javacream

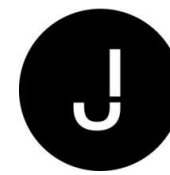
Javacream

Dr. Rainer Sawitzki

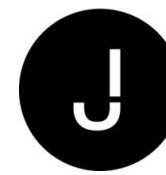
Alois-Gilg-Weg 6

81373 München

Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks, der fotomechanischen und elektronischen Wiedergabe vorbehalten.

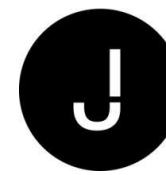


Einführung	6
Apache ServiceMix	23
Die Java Virtual Machine	35
Management mit JMX	107
Enterprise Integration Patterns	117
RESTful WebServices	136
Beispiele	150



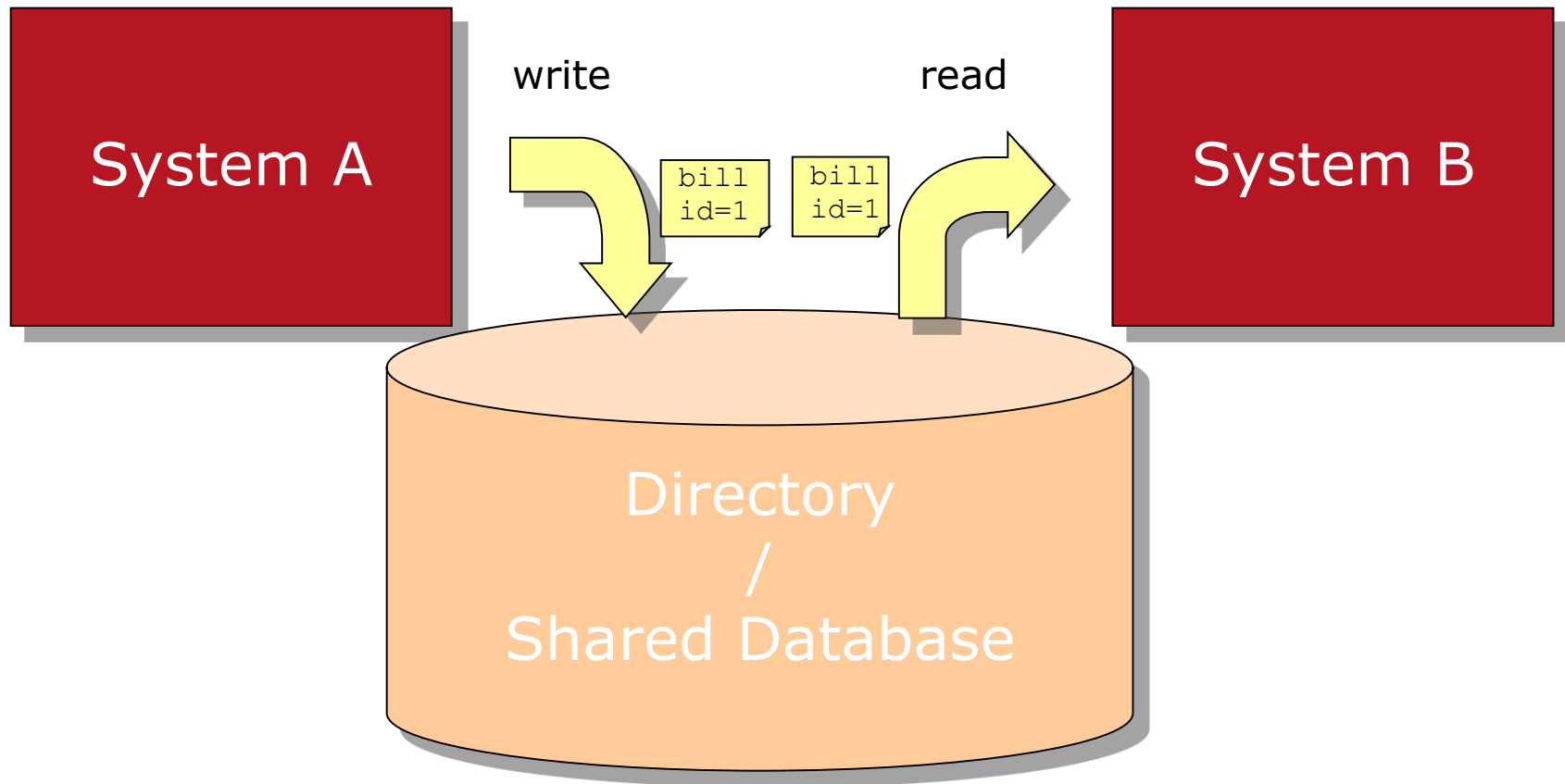
1

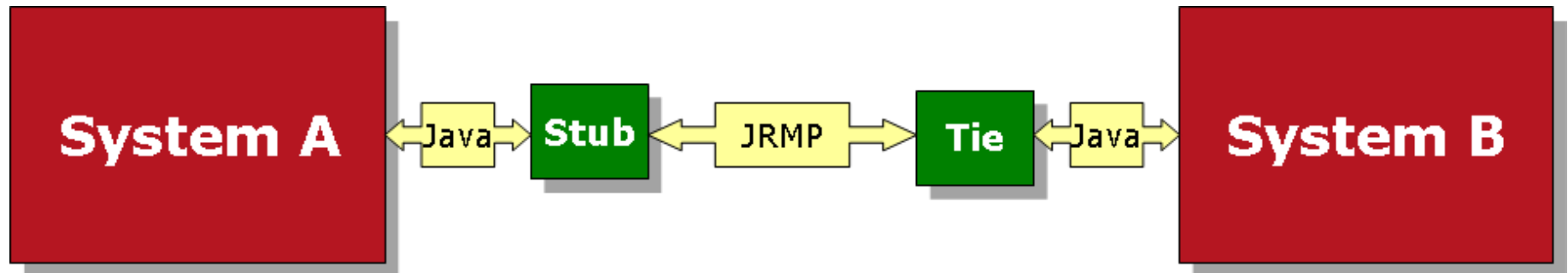
EINFÜHRUNG

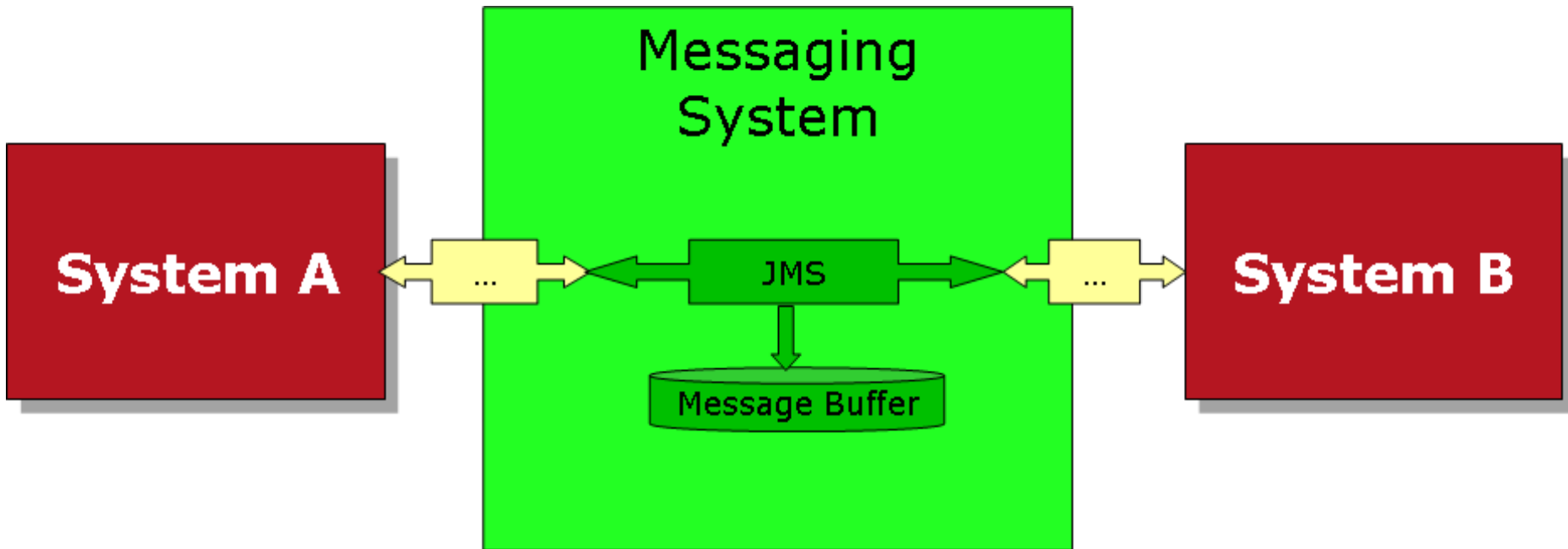


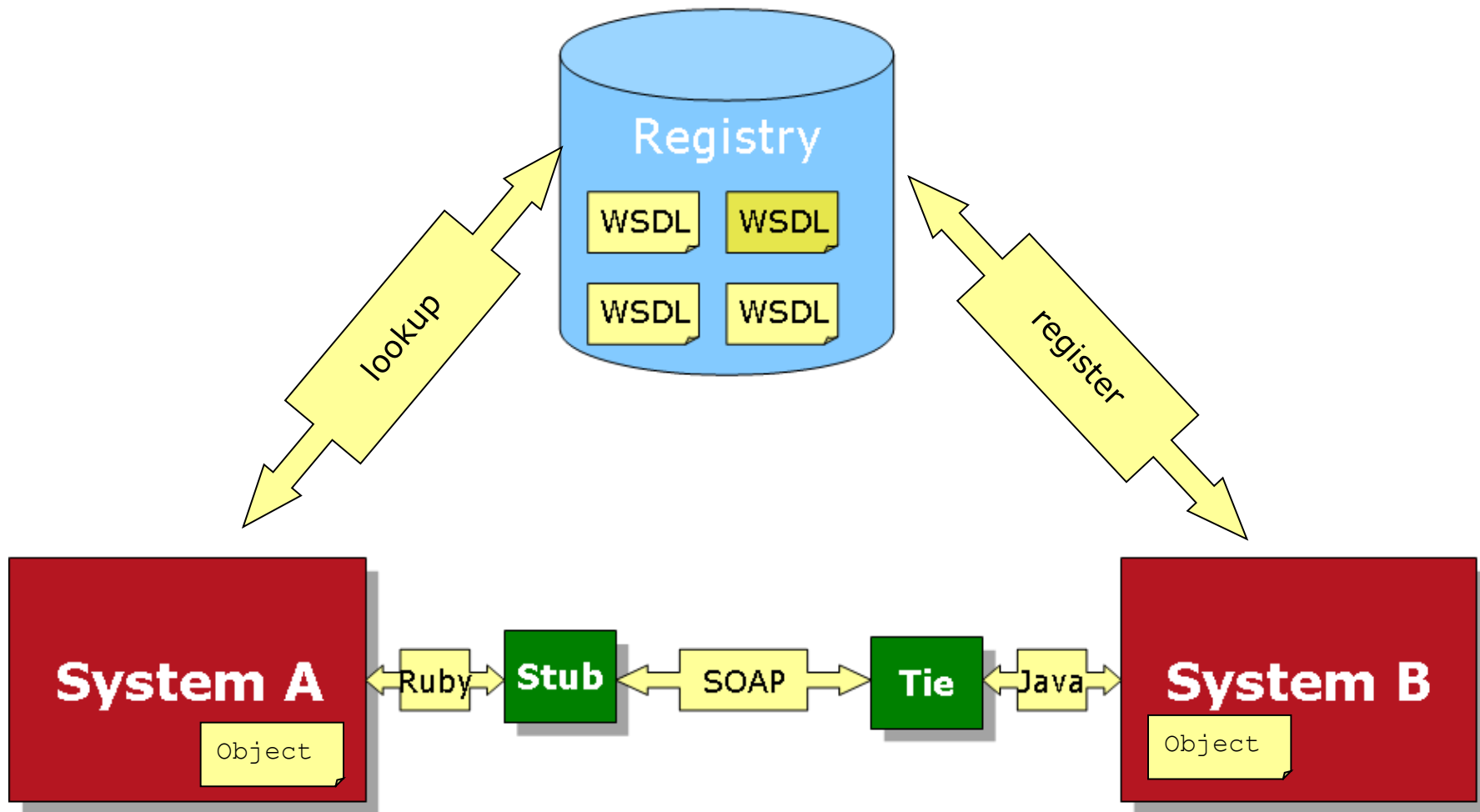
1.1

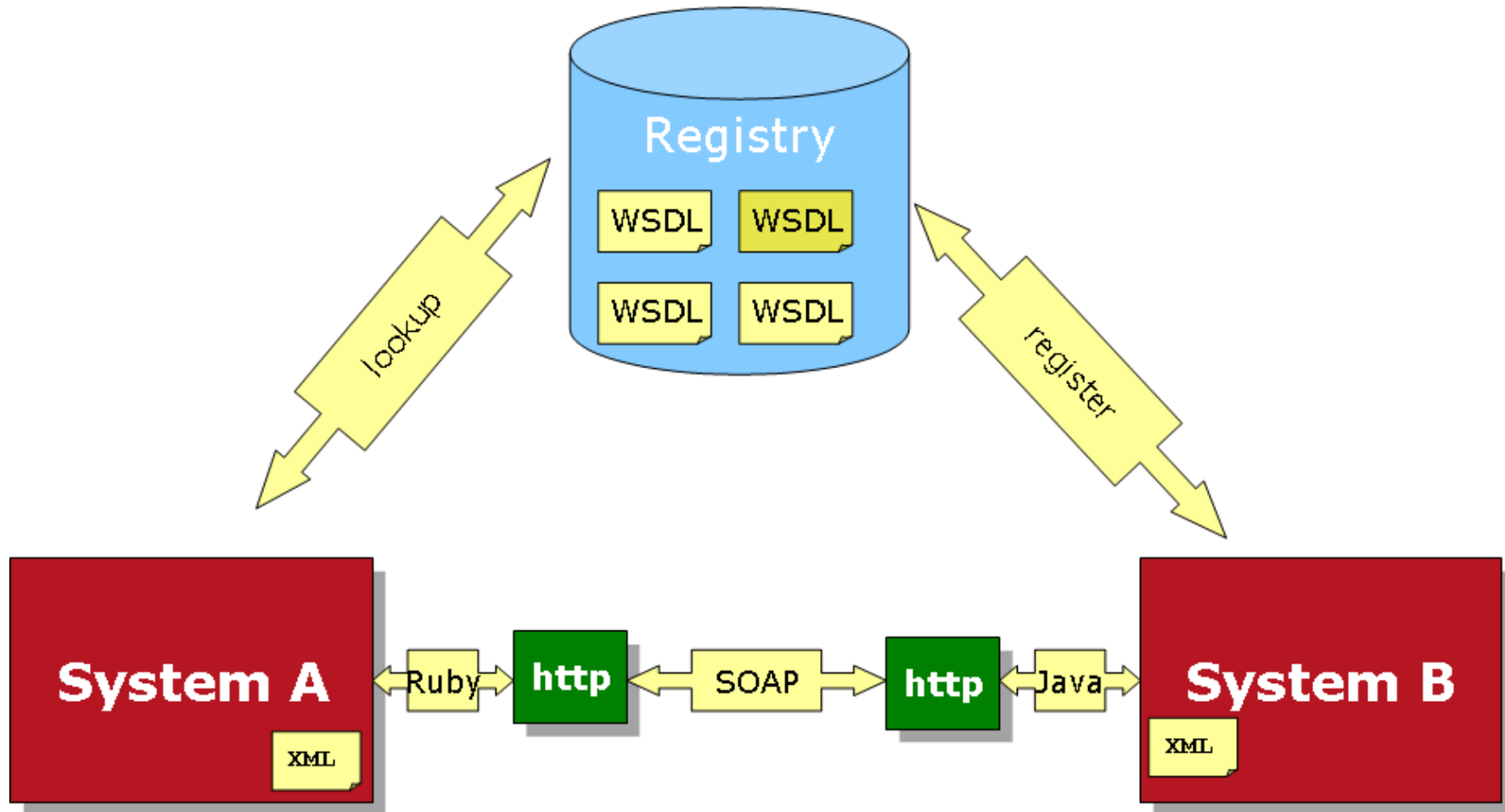
VON DER VERTEILTEN ANWENDUNG ZUM ESB

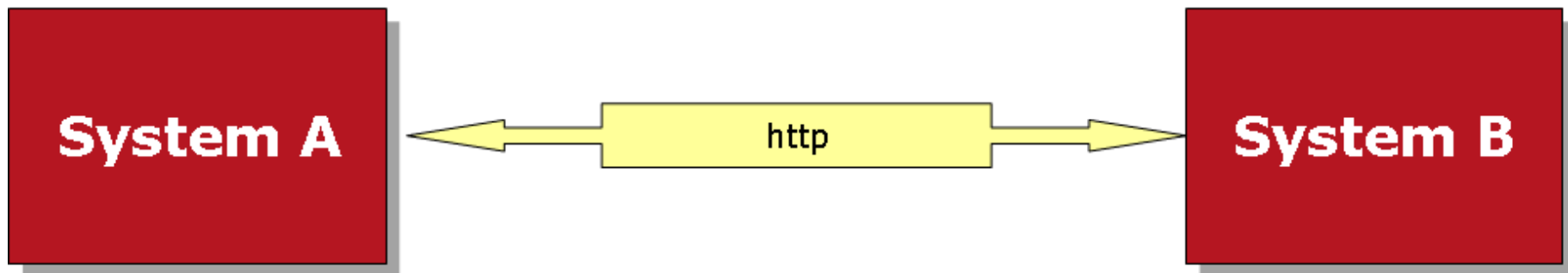
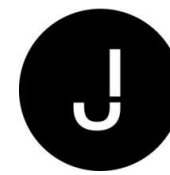


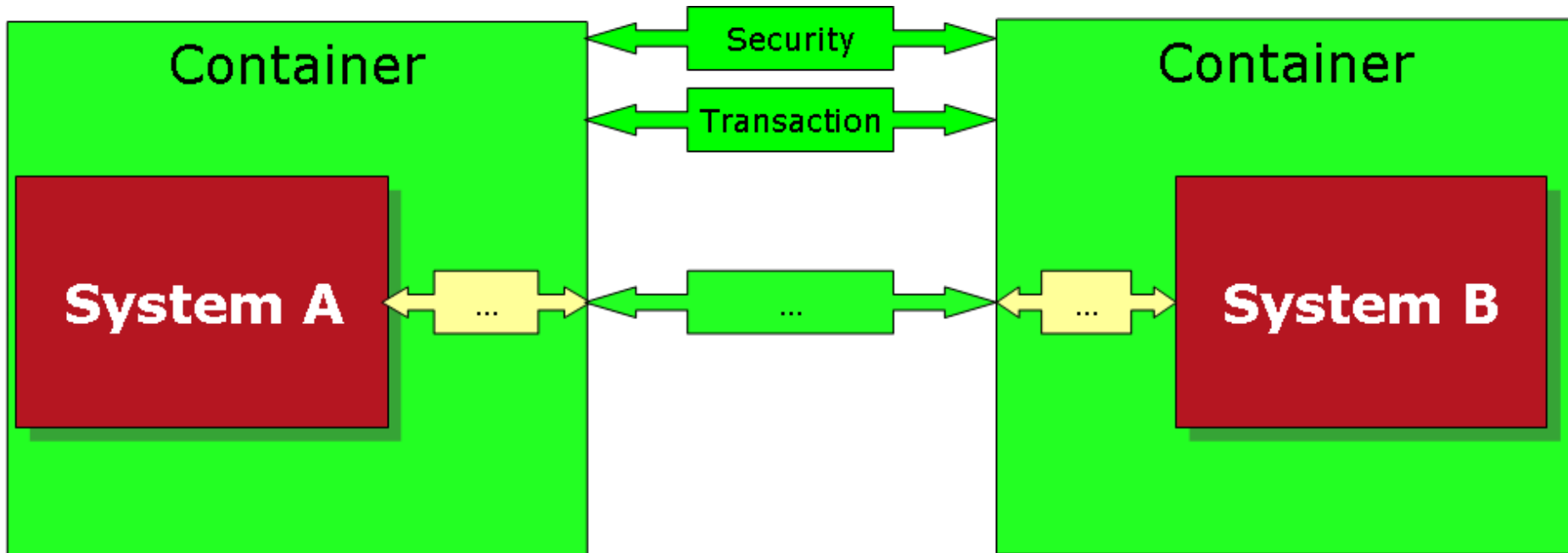


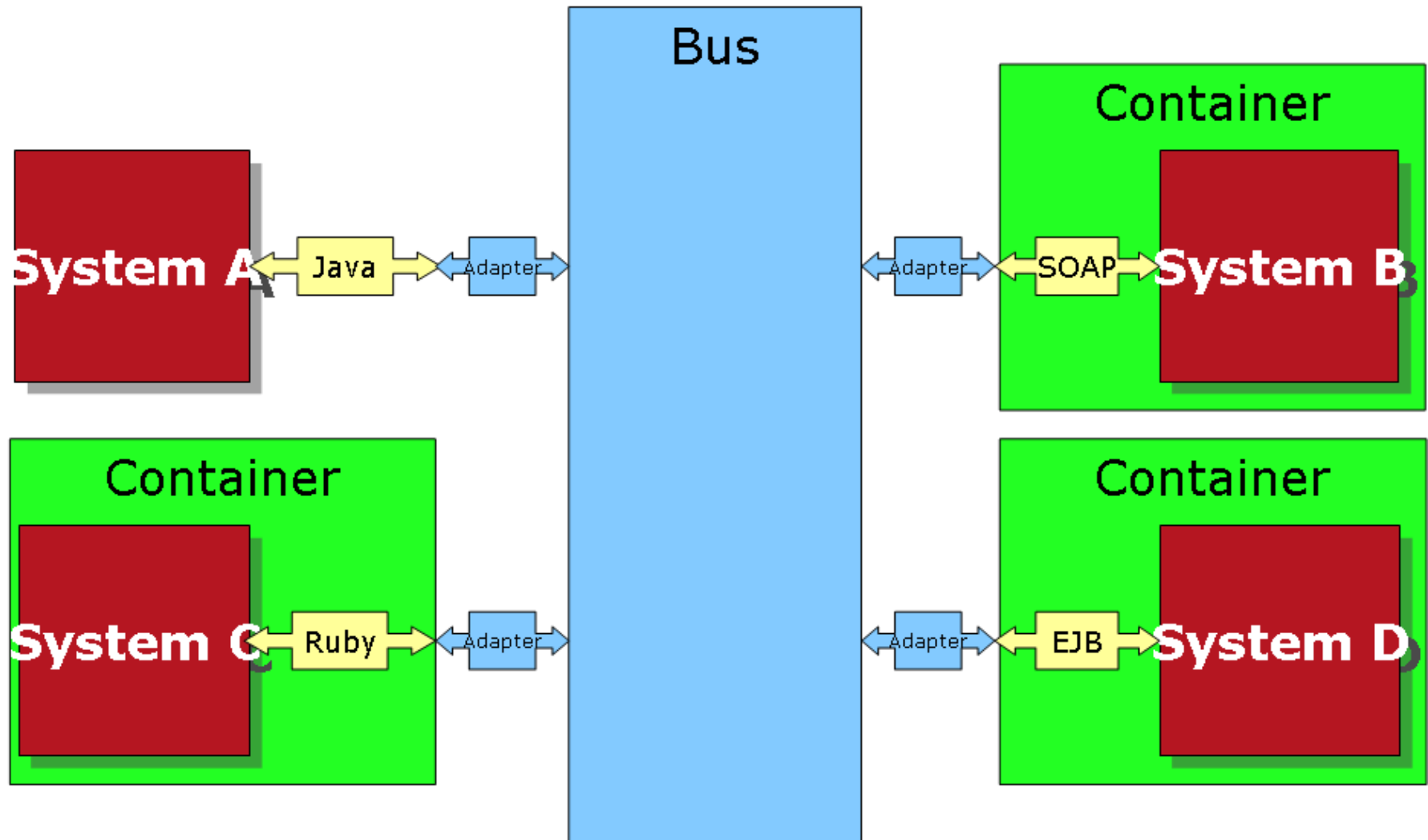


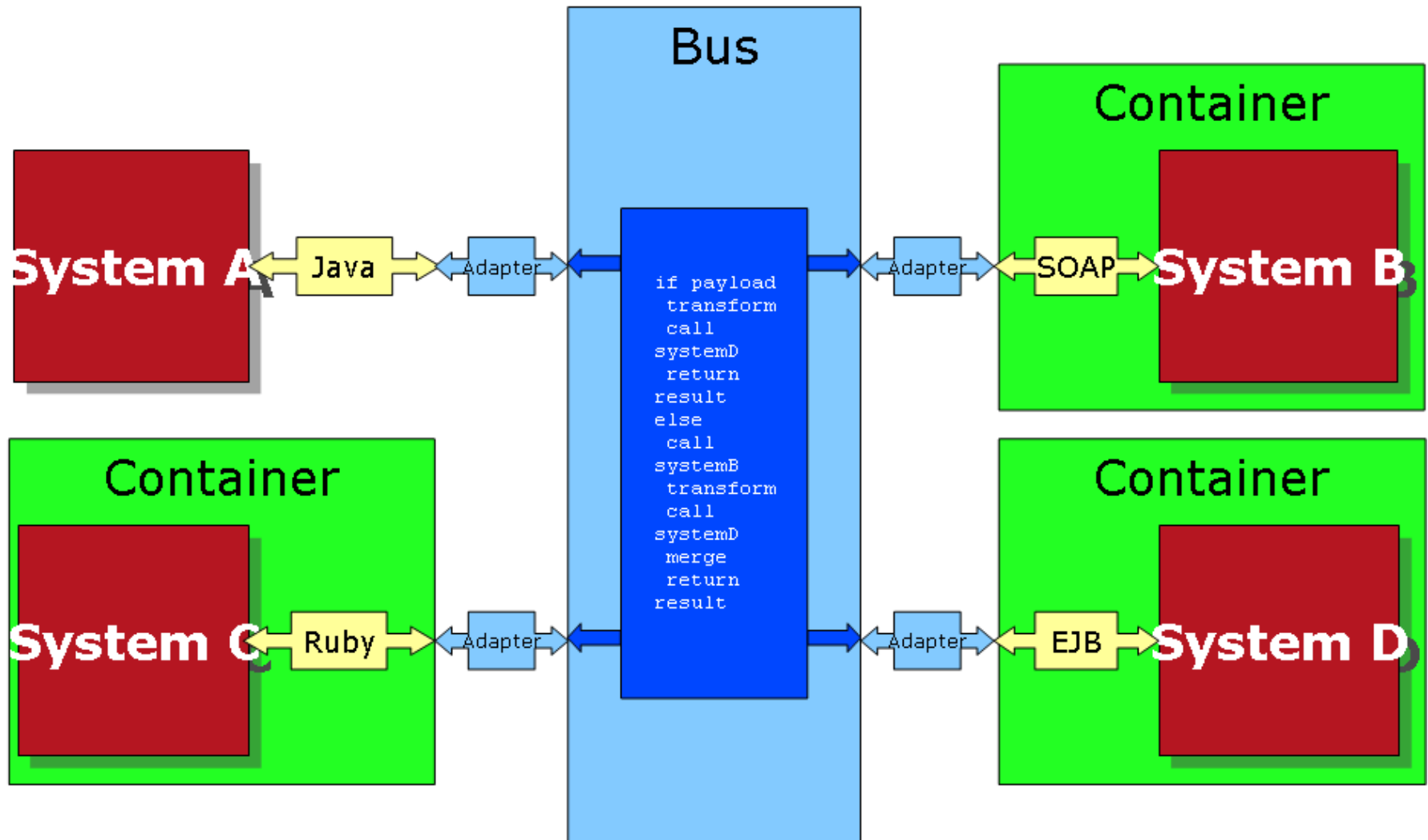


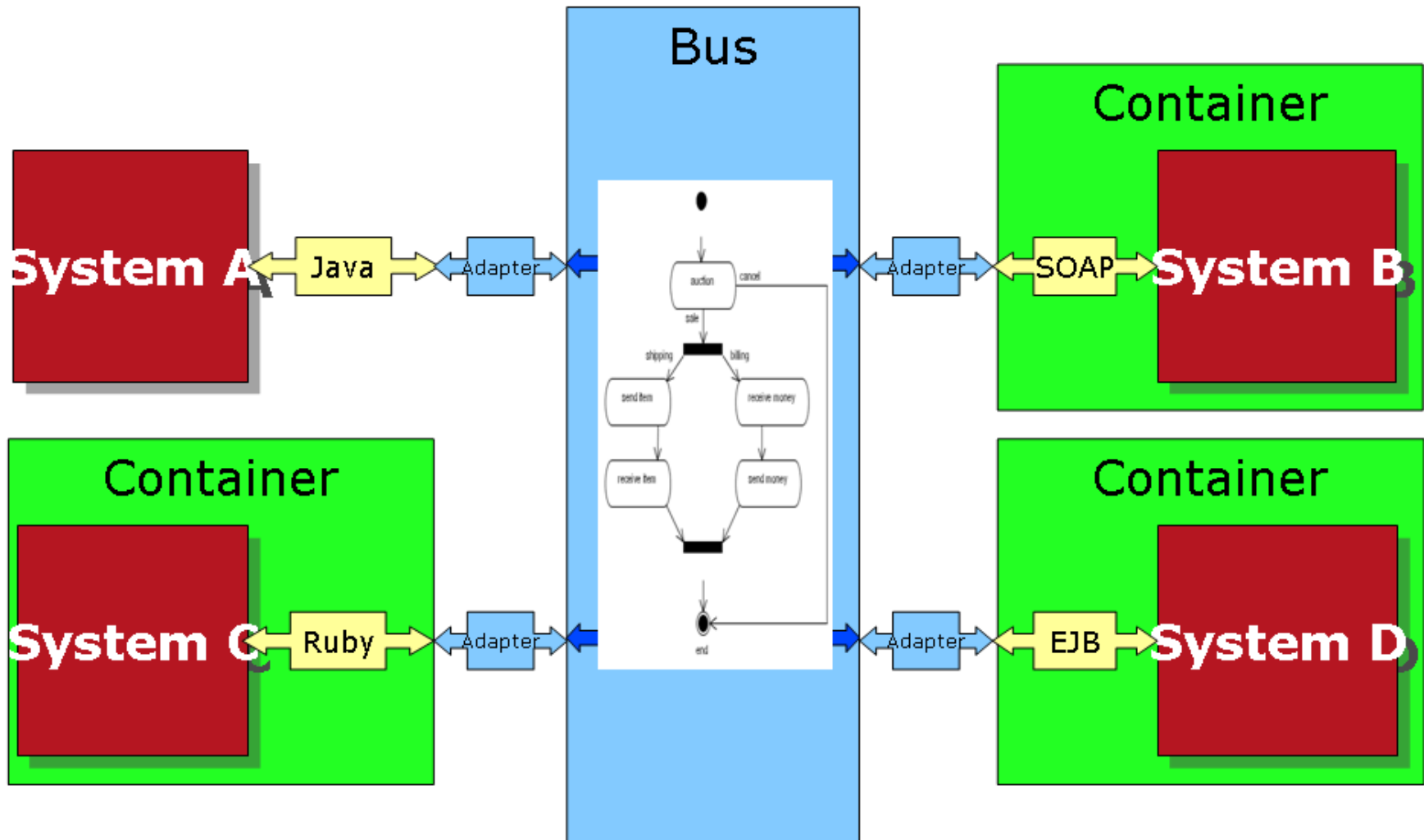




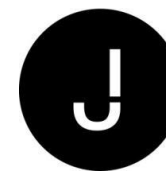








- Funktionen
 - Connectors für die gängigen Kommunikationsprotokolle.
 - Quality of service.
 - Entkopplung.
 - Laufzeitumgebung für Workflows.
- Hersteller
 - Die allermeisten aktuellen ESB-Implementierungen im Java-Umfeld beruhen auf der Applikationsserver-Technologie.
 - JBoss ESB, AquaLogic, IBM Process Manager...
 - Auch unabhängige Lösungen existieren
 - Mule
 - **Apache ServiceMix**
 - ...


















1.2

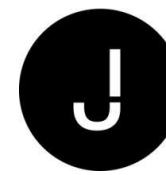
INSTALLATION UND WERKZEUGE

- Java-IDE
 - z.B. Eclipse
 - Bevorzugt: Die Spring Tools Suite (STS)
- Apache Maven
 - Dependency Management und Build-Werkzeug
 - Frei erhältlich von Apache.org
- Spring
 - Frei verfügbares Context and Dependency Injection Framework
 - Camel erweitert Spring um eine Domain Specific Language zur Routen-Definition
- Java
 - Programmiersprache zur Realisierung komplexer Geschäftslogik während der Routen-Ausführung
- JUnit
 - Ein Java-basiertes Testwerkzeug

- Entpacken der zur Verfügung gestellten ZIP-Datei enthält alle notwendigen Werkzeuge

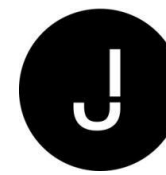
 bin
 data
 deploy
 etc
 examples
 instances
 lib
 licenses
 system
 karaf.pid
 LICENSE
 lock
 NOTICE
 README
 RELEASE-NOTES

- IDE mit allen Java-relevanten Features
 - Code Assist
 - Compiler
 - Refactoring
 - ...
- Die notwendigen abhängigen Bibliotheken
 - werden in einem Maven-POM definiert
 - von einem Build-Prozess automatisch geladen
 - und mit dem Klassenpfad des Projekts synchronisiert
- Spring Tools Suite
 - Zusätzliche PlugIns zur Verwaltung von Spring-basierten Projekten



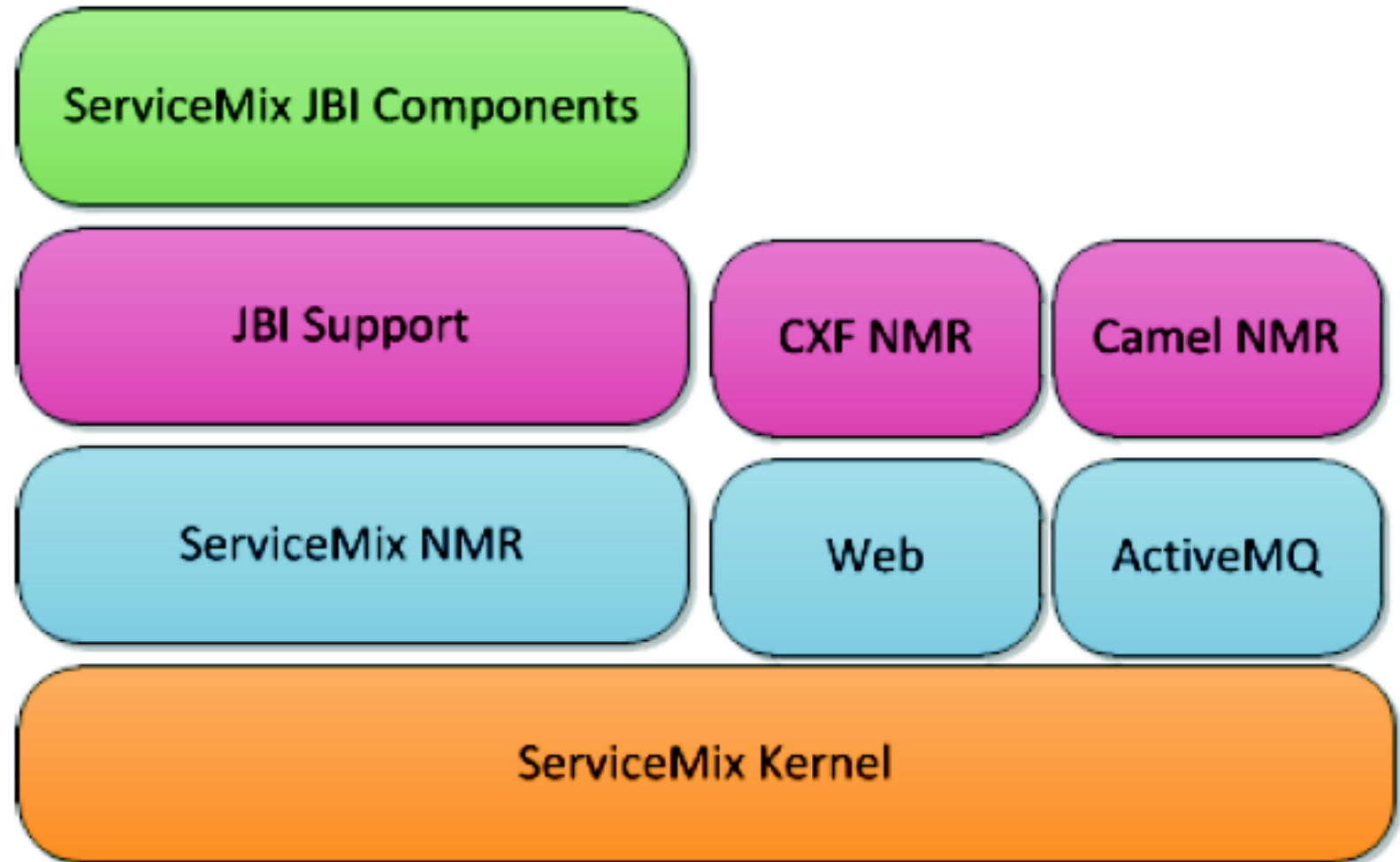
2

APACHE SERVICEMIX



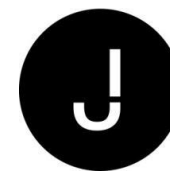
2.1

ARCHITEKTUR



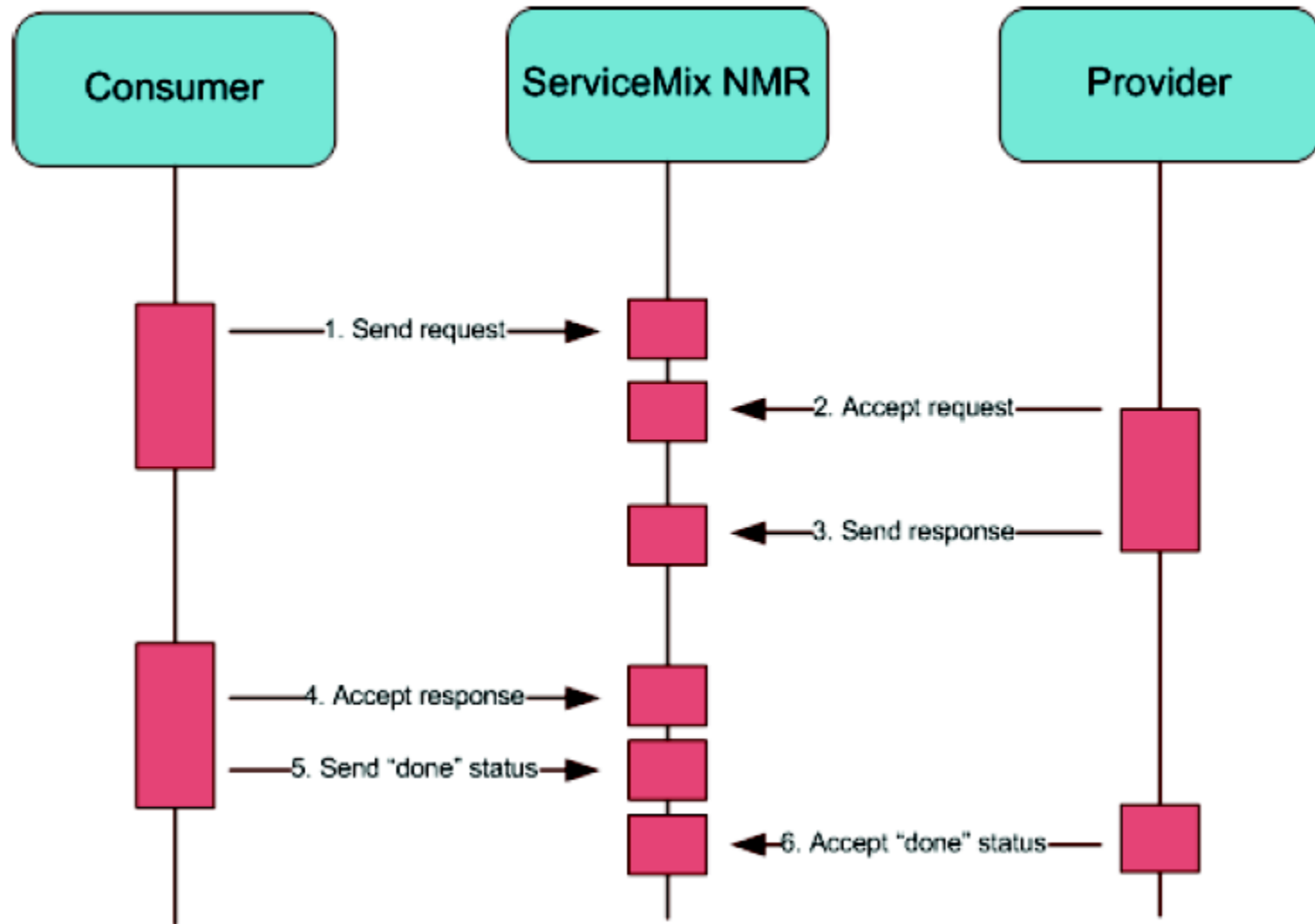
- ämtliche Nachrichten innerhalb des ServiceMix werden in einem einheitlichen Format verwaltet und übertragen
- JBI
 - Java Business Integration,
 - ein Framework, um ESB-Komponenten Hersteller-unabhängig erstellen zu können
 - Heute kaum noch relevant
- CXF
 - Ein Framework zur Definition Java-basierter Web Services
 - SOAP
 - REST
- Camel
 - Ein Framework zum Routing von Messages über verschiedene Connectors
 - Die zentrale Komponente von ServiceMix!
- ActiveMQ
 - Ein Messaging Systems mit Queues und Topics

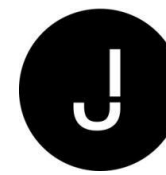
Prinzipielle Arbeitsweise (Quelle: FuseSource)



JAVACREAM

Training
Consulting
Projectmanagement





2.2

APACHE KAFKA

- Kafka ist eine Implementierung des OSGi-Frameworks
 - OSGi verwaltet so genannte "Bundles", die bestimmte Dienste zur Verfügung stellen
 - Alle ServiceMix-Komponenten sind Bundles
 - Kafka verwaltet diese Bundles
 - Dynamisches Starten und Stoppen
 - Bei Bedarf dynamische Aktualisierung
 - Bei Bedarf Hot Deployment
- Apache ServiceMix-Anwendungen werden intern immer als Bundles umgesetzt

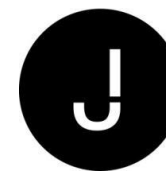
- Nach dem Start von ServiceMix wird automatisch die Kafka-Konsole gestartet
- Diese ermöglichen das Absetzen von Kafka-Kommandos
 - die dann von der Kafka-Runtime umgesetzt werden
- Beispiele:
 - Bundles
 - `list`
 - `start`
 - `stop`
 - Features (Gruppen von Bundles, die ein komplexes System realisieren)
 - `feature:list`
 - `feature:install`
 - System
 - `version`
 - `shutdown`

Die Kafka-Konsole nach dem Start von ServiceMix

[illegible]

Die Kafka-Konsole nach dem Start von ServiceMix

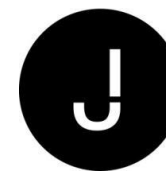
[illegible]



2.3

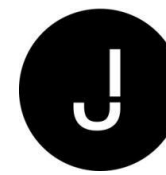
WICHTIGE VERZEICHNISSE

- bin
 - Skripte zum Starten und Stoppen von ServiceMix
- data
 - Arbeitsverzeichnis
- deploy
 - Alle Elemente in diesem Verzeichnis werden als Anwendungen installiert
- etc
 - Konfigurationsdateien
 - vorwiegend XML und Properties-Dateien
 - am Besten über STS betrachten



3

DIE JAVA VIRTUAL MACHINE



3.1

EINFÜHRUNG

- Java Applikationsserver sind zumindest in großen Teilen in der Programmiersprache Java implementiert
- Für das Verständnis der Arbeitsweise eines Applikationsservers sowie erste elementare Administrationsaufgaben (nämlich die Administration des Java Prozesses selber) ist es notwendig, sich einen zumindest ein grundsätzliches Verständnis der Arbeitsweise einer Java Anwendung notwendig
- Ein Java-Programm wird nicht direkt auf dem Betriebssystem ausgeführt
 - Stattdessen wird ein eigener Prozess gestartet, die Java Virtuelle Maschine (JVM oder VM)
 - Die VM führt dann die eigentliche Java-Anwendung aus, die dadurch selber Plattform-unabhängig sind
 - Alle Hardware- und Betriebssystem-Spezialitäten werden von der VM komplett gekapselt

- Für jede Plattformen, die Java unterstützen will, muss eine eigene Virtuelle Maschine zur Verfügung gestellt werden
- Die konkrete Implementierung erfolgt dabei natürlich durch plattformabhängige Programme

Java Binaries (Beispiel Windows)



JAVACREAM

Training
Consulting
Projectmanagement

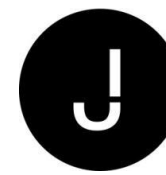
- classic
- java.exe
- policytool.exe
- tnameserv.exe
- awt.dll
- fontmanager.dll
- ioser.dll
- jawt.dll
- jpeg.dll
- jsound.dll
- NPJava11.dll
- NPJava32.dll
- verify.dll
- pluginctl130_02.cpl

- hotspot
- javaw.exe
- rmid.exe
- ActPanel.dll
- cmm.dll
- hpi.dll
- ioser12.dll
- jcov.dll
- jpins32.dll
- msvcrt.dll
- NPJava12.dll
- NPOJI600.dll
- zip.dll

- beans.ocx
- keytool.exe
- rmiregistry.exe
- agent.dll
- dcpr.dll
- hprof.dll
- java.dll
- JdbcOdbc.dll
- jpishare.dll
- net.dll
- NPJava130_02.dll
- packager.dll
- pluginctl.cpl

- Oracle spezifiziert die Virtuelle Maschine und stellt ein Verfahren zum Testen einer konkreten Implementierung zur Verfügung
 - Die Spezifikation ist frei einsehbar, z. B. über <http://java.sun.com/docs/books/vmspec>
 - Das heißt, Oracle bietet somit nicht nur ein Produkt „Virtual Machine“ an, sondern lizenziert und zertifiziert durch die JavaTest-Suite auch Anbieter, die die Spezifikation in eigenen Lösungen erfüllen
 - Dadurch konkurrieren bereits eine ganze Reihe Virtueller Maschinen für die unterschiedlichsten Plattformen, die jeweils bestimmte Aufgabenbereiche besonders effizient abdecken können

- Die Produktpalette Virtueller Maschinen ist sehr umfassend
 - Es existieren spezielle Ausprägungen für Server, PC-Systeme und Kleingeräte, aber auch Maschinen, die zur Laufzeit die Ausführungsgeschwindigkeit der Programme optimieren können
- Oracles Virtuelle Maschine beinhaltet die „HotSpot“-Technologie zur Performance-Steigerung
 - Werkzeuge zum Testen von Java-Programmen benötigen spezielle Routinen zum automatischen Sammeln der Profiling-Information zur Ausführungszeit
- Der Administrator einer Java-Anwendung kann somit aus verschiedenen konkurrierenden Produkten das am besten geeignete wählen
 - Der Java-Programmierer sieht aber nur die allgemein gültige Schnittstelle der Spezifikation und kann sich auf Grund der Zertifizierung von Oracle auf eine korrekte Implementierung der Spezifikation verlassen



3.2

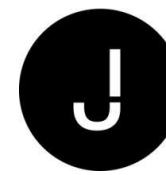
JAVA PROGRAMMIERUNG

- Jede Java-Anwendung ist in relativ kleine Module unterteilt, die so genannten Klassen
 - Diese Klassen haben Standardmäßig die Endung „.class“
- Nachdem ein Applikationsserver ein komplexe Java-Anwendung ist und zusätzlich noch die Klassen der Anwendungen kommen, die innerhalb des Servers installiert sind, kann die Virtuelle Maschine auch gezippte Verzeichnisbäume lesen
 - Diese Dateien, die Java-Archive haben die Endung „.jar“ (Englisch für Gefäß) und enthalten beliebig viele Klassen
- Die Virtuelle Maschine kann nun nicht nur ein Java-Archiv lesen sondern wiederum eine beliebige Menge davon

- Die hierarchische Paketstruktur erfordert zur eindeutigen Identifikation einer Java-Klasse nicht nur den Namen der Klasse, sondern auch die Angabe des Paketes
- Dies erfolgt durch den voll qualifizierten Klassennamen, der den Punkt als Trennzeichen verwendet
- Konventionen:
 - Die Systemklassen befinden sich entweder im Paket java oder in javax
 - Paketnamen bestehen aus Kleinbuchstaben
 - Klassennamen beginnen mit einem Großbuchstaben
 - Alle anderen Klassen sollen in eine Paketstruktur integriert sein, deren beide Hauptpakete mit der „umgedrehten“ WWW-Adresse beginnen

- Die Virtuelle Maschine lädt nur bestimmte Klassen direkt in den Arbeitsspeicher
- Diese so genannten „Bootstrap-Klassen“ sind die Systemklassen, die in der Datei rt.jar im lib-Verzeichnis gefunden werden
- Ein weiteres Java-Archiv, i18n.jar enthält ebenfalls Systemklassen
- Eine besondere Systemklasse ist der so genannte Klassenlader
 - Dieser ist in der Lage, eine frei wählbare Anzahl von Lokationen nach Java-Archiven und Java-Klassen zu durchsuchen und der Virtuellen Maschine zur Verfügung zu stellen
 - Um die Anzahl und Ort der zu durchsuchenden Quellen frei einstellen zu können, verwendet der Standard-Klassenlader die Umgebungsvariable CLASSPATH

- Dieser Pfad enthält eine über Semikolon (Windows) oder Doppelpunkt (Solaris) getrennte Liste von Java-Archivdateien und Verzeichnissen
- Die Virtuelle Maschine versucht nun, eine in einem Programm verwendete Klasse dadurch zu laden, indem
 - als erstes die Systemklassen durchsucht werden und, falls die angesprochene Klasse darin nicht gefunden wird,
 - danach über den ClassLoader der Klassenpfad in der angegebenen Reihenfolge
- Wird die benötigte Klasse gefunden, wird sie einmalig in den Speicher geladen
 - Sonst tritt ein Fehler auf
- Der Klassenpfad kann sowohl als Umgebungsvariable gesetzt werden als auch der Virtuellen Maschine beim Programmstart als Option (-classpath bzw. -cp) mitgegeben werden

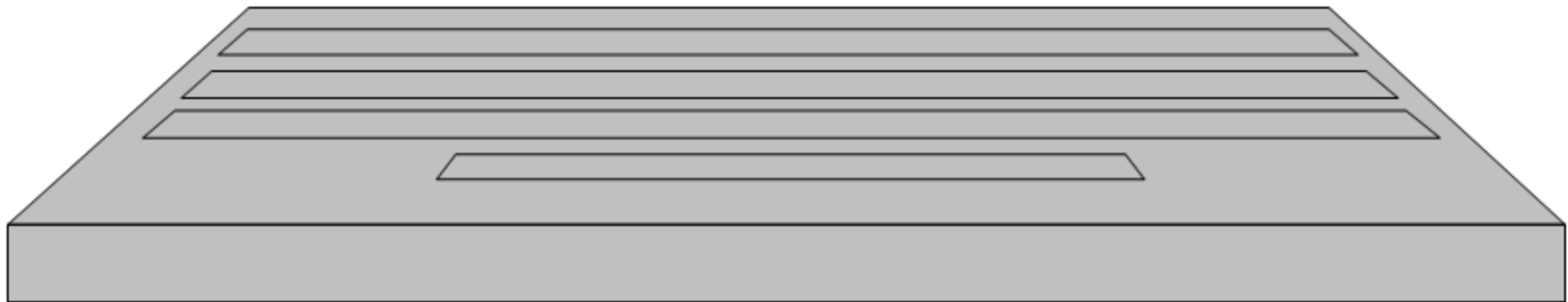


3.3

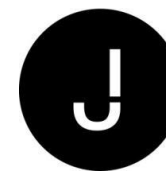
FEHLERPROTOKOLLE

- Tritt während eines Programms eine Fehlersituation auf, so wird von der Anwendung ein „Stack Trace“ geschrieben
- Diese Stack Traces haben in Java-Anwendungen einen bestimmten Aufbau:
 - Der Fehlertyp wird durch eine Java-Klasse, die „Exception“, definiert, deren voll qualifizierter Klassenname den Stack Trace einleitet
 - Es folgt, falls vorhanden, ein mehr oder weniger sprechender Fehlerbeschreibungstext
 - Als drittes erfolgt der eigentliche Stack Trace, der den gesamten Aufruf, also den Method Frame, ausgibt
 - Ist die Anwendung auf spezielle Art und Weise erstellt worden, werden hierbei sogar noch die Zeilennummern des Quellcodes mit ausgegeben
 - Für die Fehlersuche für den Entwickler eine unschätzbare Hilfe


```
java.net.SocketException: error setting
options
    at
    java.net.PlainDatagramSocketImpl.join(Nat
ive Method)
    at
    java.net.PlainDatagramSocketImpl.join(Pla
inDatagramSocketImpl.java:134)
    at
    java.net.MulticastSocket.joinGroup(Multic
astSocket.java:274)
```



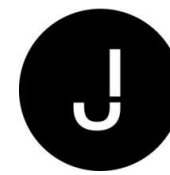
- Die Interpretation des Stack Traces ist in der Praxis häufig jedoch nicht so einfach:
 - Fehlende beschreibende Texte
 - Vielfache Ausgabe von Fehlermeldungen, da der Stack Trace im Rahmen des Programms an mehreren Stellen ausgegeben wird
 - Schwer zu lesende Ausgaben durch verschachtelte Exceptions
 - Bei verteilten Anwendungen ist die Frage, auf welcher Maschine der eigentliche Fehler protokolliert wurde
 - Eine unspezifische „java.rmi.RemoteException“ auf Client-Seite kann beispielsweise auf dem Server als „java.sql.SQLException: Duplicate Key“ eine klare Ursache haben
- Insgesamt erfordert die Interpretation von Stack Traces einiges an Erfahrung
 - Wichtig für den Administrator eines Systems ist es dabei, Stack Traces möglichst vollständig zu protokollieren und damit aussagekräftig an den Entwickler weiterleiten zu können



3.4

AUFRUF EINES JAVA-PROGRAMMS

- Der Aufruf einer Java-Anwendung erfolgt durch den Aufruf des ausführbaren Java-Prozesses namens „java“
 - Falls dieses Programm nicht im Pfad für ausführbare Dateien des Betriebssystems gefunden wird, kann der Aufruf auch mit vollständiger Pfadangabe erfolgen
- Um die Installation der Java-Laufzeitumgebung zu testen, kann der Aufruf mit der Option „-version“ erfolgen



```
C:\WINDOWS\system32\cmd.exe

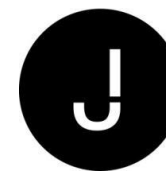
Z:\>java -version
java version "1.5.0_06"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_06-b05)
Java HotSpot(TM) Client VM (build 1.5.0_06-b05, mixed mode, sharing)

Z:\>_
```

- Nachdem auf einer Rechner-Installation auch mehrere Java Laufzeitumgebungen in verschiedenen Versionen parallel installiert sein können gehört dieser Test beim Auftreten „merkwürdiger“ Fehler zum Standard-Fehlersuche
- So zeigt die folgende Fehlermeldung an, dass eine Java-Anwendung von einer Virtuellen Maschine ausgeführt werden soll, die eine zu niedrige Version aufweist:
- Exception in thread "main" java.lang.UnsupportedClassVersionError: org/javacream/books/client/BooksClient (Unsupported major.minor version 49.0)

- Der Klassenpfad, in dem alle Klassen gefunden werden, die die Anwendung ausmachen, wird durch die „classpath“-Option gesetzt
- Auch hier können natürlich Fehler auftreten:
 - `java.lang.ClassNotFoundException <Klassenname>`
 - `java.lang.NoClassDefFoundError <Klassenname>`
 - `java.lang.LinkageError <Klassenname>`
- Die ersten beiden Exception-Ausgaben sind relativ einfach zu interpretieren
 - Es wird eine Klasse benötigt, die im gesamten Klassenpfad nicht zu finden ist
 - Der Unterschied ist nur für einen Java-Programmierer relevant

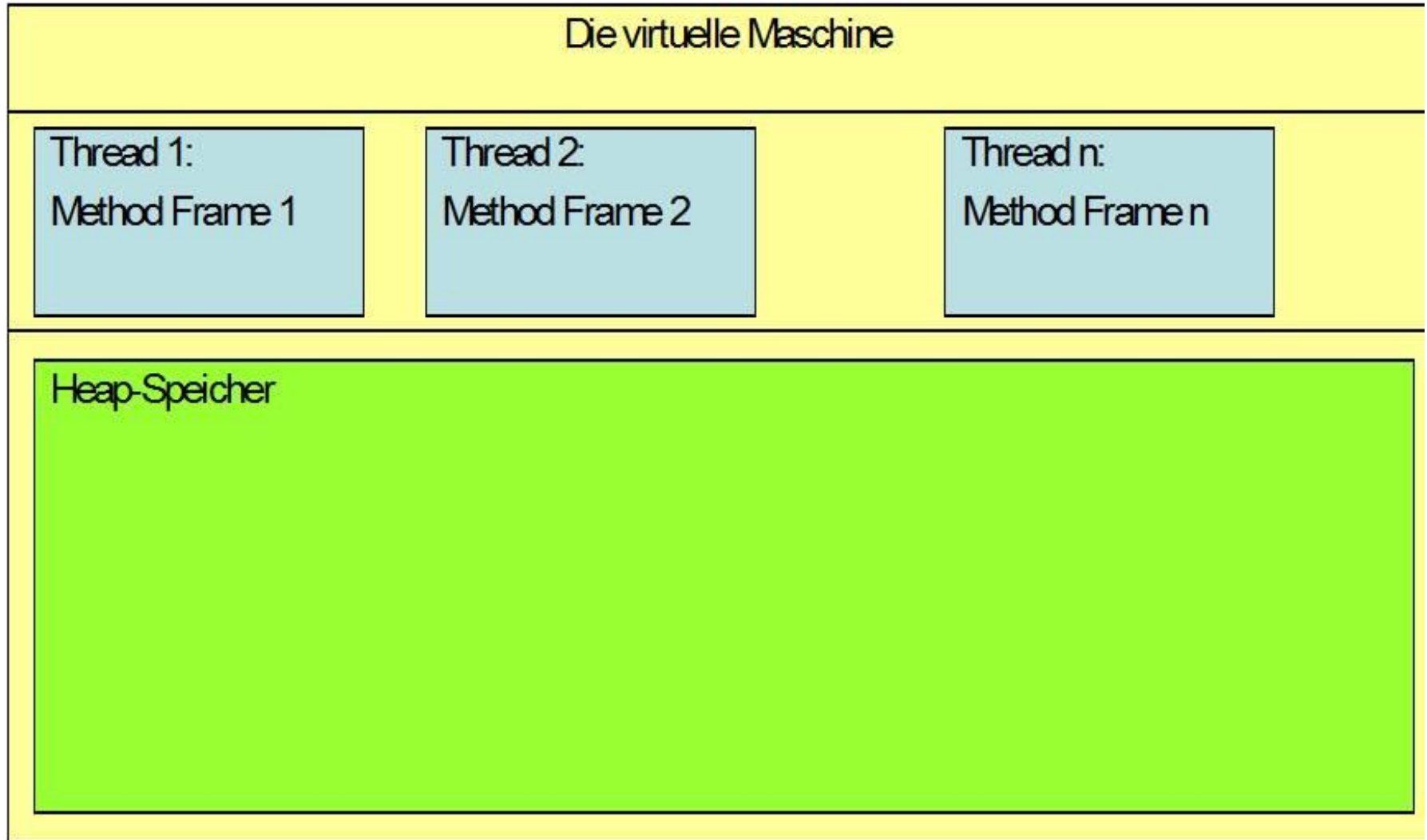
- Die dritte Fehlermeldung ist hingegen recht kritisch:
 - Sie zeigt an, dass die Anwendung eine Klasse sehr wohl findet, nun aber mehrfach in unterschiedlichen Versionen
 - Ein Applikationsserver benutzt viele verschiedene Klassenlader und damit kann es sehr wohl zu dieser Fehlersituation kommen
- Die Behebung des Fehlers kann aller Erfahrung nach jedoch nicht vom Administrator des Applikationsservers alleine übernommen werden
 - Der Fehler liegt mit höchster Wahrscheinlichkeit an den zu installierenden Anwendungsprogrammen bzw. den erzeugten Java-Archiven



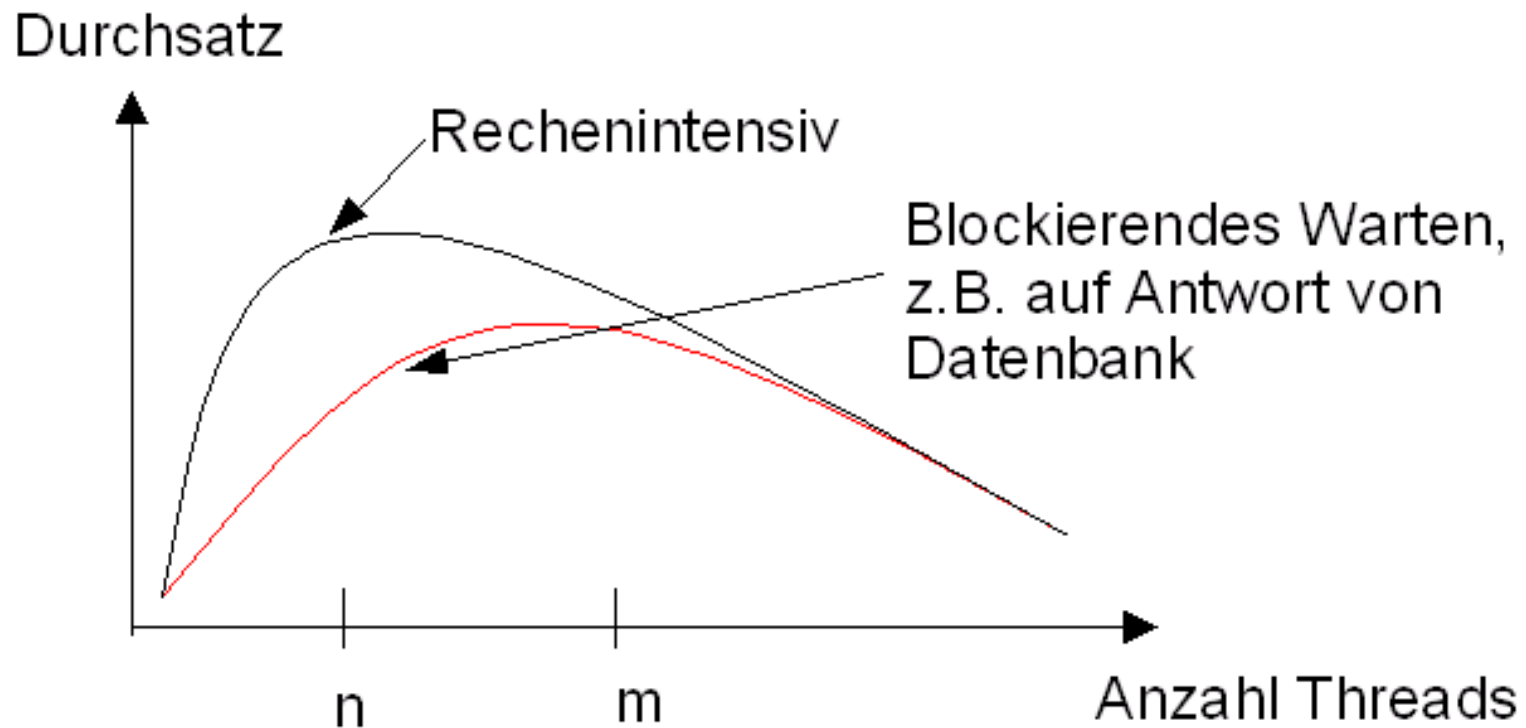
3.5

THREADS

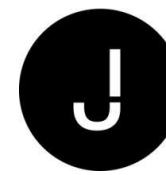
- Die Virtuelle Maschine führt nun ihre Aufgaben nicht in einem einzigen Prozess seriell durch sondern verwendet dafür verschiedene Threads
- Viele (auf Server-Seite sind mehrere Hundert Threads keine Seltenheit) Threads laufen innerhalb der Virtuellen Maschine „parallel“ und können so beispielsweise auf einer Mehrprozessormaschine tatsächlich nebenläufig ausgeführt werden
- Jeder dieser Threads hat einen eigenen Stack-Bereich, den so genannten „Method Frame“ für seine aktuellen Methodenaufrufe, lokalen Parameter etc.



- Threads sind für eine Virtuelle Maschine relativ aufwändig zu verwalten
 - Die Menge an Threads, die innerhalb eines Java-Prozesses effizient parallel ablaufen können, hängt natürlich stark von der verwendeten Hardware ab
- Daneben gibt es aber auch einige Regeln und Kriterien, die für die optimale Einstellung relevant sind
 - So verwaltet ja wie bereits erwähnt ein Applikationsserver einen oder mehrere Thread Pools um eingehende Requests bearbeiten zu können



- Bei einer rein Rechenintensiven Anwendung, die ausschließlich auf dem Applikationsserver läuft ist der optimale Durchsatz schnell bestimmt:
 - Nachdem jeder Thread-Wechsel selber wiederum Prozessorleistung benötigt liegt das Maximum bei der Anzahl der Prozessoren
- Nicht so einfach (und leider auch eher die Realität) sind Anwendungen, die zumindest zeitweise auf Antwort eines anderen Systems warten und während dieser Wartezeit den aufrufenden Thread innerhalb des Applikationsservers blockieren
 - Nun kann die Menge der konfigurierten Threads selbstverständlich die Anzahl der Prozessoren deutlich übersteigen
 - Der Optimalwert m ist die Anzahl von Threads, bei der zum ersten mal 100% Auslastung erreicht werden
 - Der Umkehrschluss „100% Prozessorlast = Optimale Konfiguration“ ist allerdings falsch: Ein zu groß dimensionierter Thread Pool führt durch überflüssige Threadwechsel ebenfalls zur Auslastung der Prozessoren, ohne dass Anfragen abgearbeitet werden



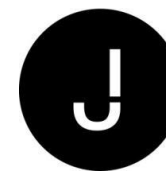
3.6

HEAP-SPEICHER

- Der Prozess „Virtuelle Maschine“ verwaltet den ihm zugeteilten Speicher und Prozessorzeit in einer eigenen, wohldurchdachten Art und Weise
- Der Speicher der virtuellen Maschine heißt „Heap“ und entspricht im Wesentlichen dem zugeteilten Hauptspeicher
- Darin werden Informationen in Form von Objekten (in etwa frei definierbaren Strukturen) und Arrays (Listen von Objekten und Datentypen) abgelegt
- Die Größe des Heap-Speichers wird beim Programmstart der Virtuellen Maschine über die Optionen
 - -Xmxn
 - Maximaler Heap-Speicher
 - -Xmsn
 - Startgröße des Heap-Speichers

- Die aktuelle Größe des Heap-Speichers und die reale Größe des VM-Prozesses können sich unterscheiden, da die Größe zur Laufzeit intern angepasst werden kann ohne frei gewordenen Speicher sofort dem Betriebssystem zurück zu geben
 - Werkzeuge, die einfach die für den Prozess „Virtuelle Maschine“ reservierten Speicher messen, sind für Monitoring und Analyse deshalb vollkommen ungeeignet
- Auch ist die aktuelle Größe des Heap-Speichers nur bedingt ein Kriterium für den aktuellen Speicherbedarf einer installierten Anwendung:
 - Wird ein Server beispielsweise gestartet mit `java -Xms256m -Xmx256m` so ist der Heap immer 256 Megabyte groß, obwohl vielleicht noch gar keine Anwendung installiert wurde und deshalb „eigentlich“ kein Speicherbedarf vorhanden ist

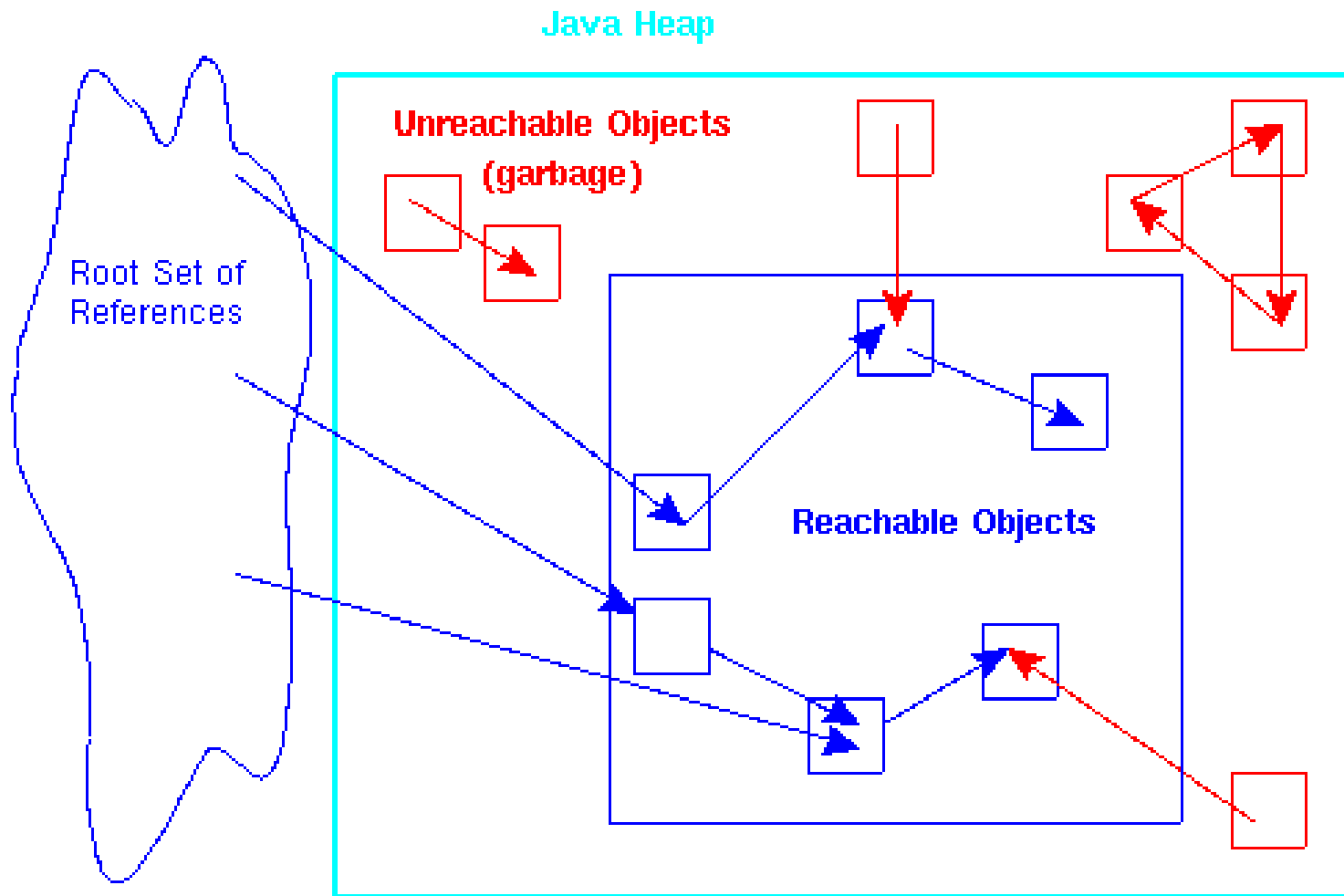
- Benötigt ein Java-Prozess mehr Speicher als ihm beim Aufruf zugestanden wurde, wird ein `java.lang.OutOfMemoryError` geworfen
- Das Auftreten dieses Fehlers führt jedoch nicht zu einer Beendigung der Virtuellen Maschine:
 - Kann die VM nach Auftreten des Fehlers wieder Speicher freigeben, so arbeitet sie ganz normal weiter
 - So kann beispielsweise eine Datenbankabfrage mit ungeschickten Selektionskriterien eine Unmenge von Datensätzen liefern, die den Speicher des Applikationsservers sprengt
 - Dann wird das weitere Einlesen der Daten abgebrochen, aber es werden auch die bereits gelesenen Daten verworfen und damit ist wieder Speicher frei



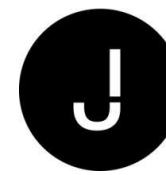
3.7

GARBAGE COLLECTION

- Ein, insbesondere für Java-Server wichtiges Feature der Virtuellen Maschine ist die Garbage Collection, die automatisch den Heap von nicht mehr zugreifbaren und damit unnötigen Objekten befreit
- Dies verhindert eine Vielzahl möglicher Programmierfehler, die zum Auftreten von Speicher-Lecks führen könnten

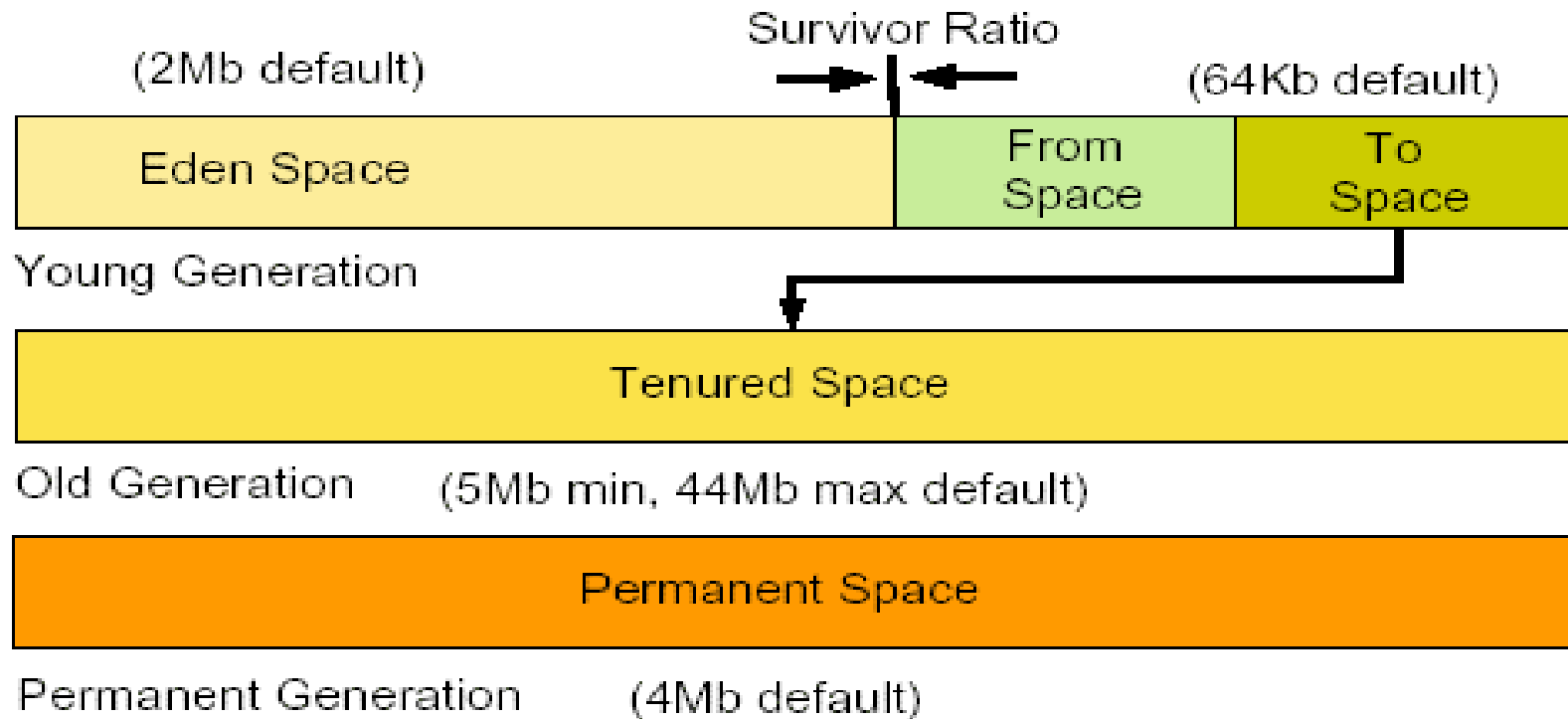


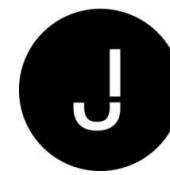
- Die korrekte Konfiguration der Garbage Collection ist insbesondere für lang laufende Java-Anwendungen im Hochlastbereich einer Server-Anwendung kritisch
- Die modernen Java-Laufzeitumgebungen bieten dafür eine Vielzahl von Optionen, die im Folgenden detailliert erläutert werden



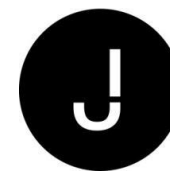
3.8

STRUKTUR DES HEAP-SPEICHERS

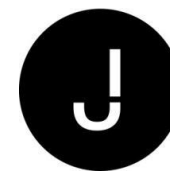




- Young Generation
 - Eden Space
 - From Space
 - To Space
- Old Generation
- Permanent Generation
 - Ab Java 8 nicht mehr im Einsatz



- Alle neu angelegten Objekte werden im Eden Space angelegt und verbleiben dort bis zur ersten Garbage Collection

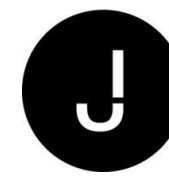


- Zwischenspeicher für Objekte, die noch gültig sind aber noch nicht alt genug für die Old Generation sind
- Objekte im Eden-Space, die von der Garbage Collection nicht entfernt werden dürfen, weil sie noch benötigt werden, werden alternierend in den jeweils nächsten Survivor Space kopiert
- Nach einer Garbage Collection in der Young Generation ist damit der Eden Space sowie ein Survivor Space leer, der andere Survivor Space hält die noch gültigen Objekte

- Länger gültige Objekte werden aus dem Survivor Space in die Tenured Generation transferiert
- Auch in der Tenured Generation läuft eine Garbage Collection ab, die diese bereinigen kann

- Klassenobjekte werden in der Permanent Generation abgelegt, da diese Objekte in der Regel niemals entladen werden
- Für die Permanent Generation wird prinzipiell überhaupt keine Garbage Collection benötigt
 - Trotzdem wird auch dieser Bereich bereinigt
 - Die Permanent Generation war eine experimentelle Idee der JVMs vor Java 8, die verworfen wurde

- Neu angelegte Objekte werden im Eden Space erzeugt
- Ein Survivor Space ist stets leer
- Der zweite wird während einer „Copy Collection“ mit den gültigen Objekte aus Eden und dem ersten befüllt
- Objekte, die genügend alt sind, werden in die Old Generation verschoben
- Der Trick bei der Konfiguration der Garbage Collection ist nun die, die Größe der Bereiche den realen Anwendungen anzupassen und somit ein „rundes“ laufen des Collectors zu garantieren

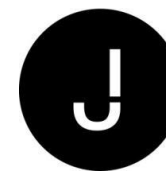


- Gibt es tatsächlich auch: Native Memory
 - Eine Art RAM-Disk
- Dafür gibt es aber noch kein "offizielles" Programmiermodell
 - Allerdings wird dies in der nächsten Sprachversion erfolgen
- Moderne Java-basierte Cache-Produkte benutzen bereits Native Memory
 - z.B. EHCache/Terracotta

- Profiling der Anwendung bezüglich des Speicherverhaltens ist insbesondere für Server-Anwendungen essenziell
- Bei Fehlkonfigurationen sind Pausenzeiten von mehreren Sekunden („Stop the World Collections“) beziehungsweise drastische Performance-Einbußen die Regel

- Mit den folgenden Einstellungen liefert die Java Virtual Machine detaillierte Informationen über das Laufzeitverhalten des Garbage Collectors:
 - `-verbose:gc`
 - `-Xrunhprof`
 - `-XX:+PrintGCDetails`
 - `-XX:+PrintGCTimeStamps`
 - `-XX:+PrintHeapAtGC`

- Es existieren aber auch eine Reihe von Werkzeugen, die die Ergebnisse in analysierbarer Form aufbereiten:
 - Eine zwar alte aber sehr schöne Oberfläche ist der GCViewer, ein Open Source Werkzeug der Tagtraum Industries
 - Als Bestandteil der Java-Installation VisualVM

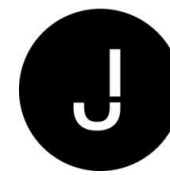


3.9

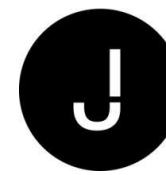
SPEICHERKONFIGURATION

- Allgemeine Einstellungen
 - `-Xms`
 - `-Xmx`
- Einstellung der Old Generation:
 - `-XX:MinHeapFreeRatio`
 - `-XX:MaxHeapFreeRatio`
- Einstellungen für die Permanent Generation
 - `-XX:PermSize`
 - `-XX:MaxPermSize`
 - `-Xnoclassgc`

- Die Garbage Collection muss nicht mehr benötigte Objekte schnell und effizient aus dem Heap-Speicher entfernen können
- Insbesondere bei großen Heap-Speichern kann die Garbage Collection erheblich Ausführungsgeschwindigkeit kosten
- Dieser Einfluss zeigt sich besonders bei Mehrprozessor-Systemen, da nicht alle Phasen der Garbage Collection parallel laufen können



- Minor Collections ohne Defragmentierung
- Major Collections mit Defragmentierung



3.10

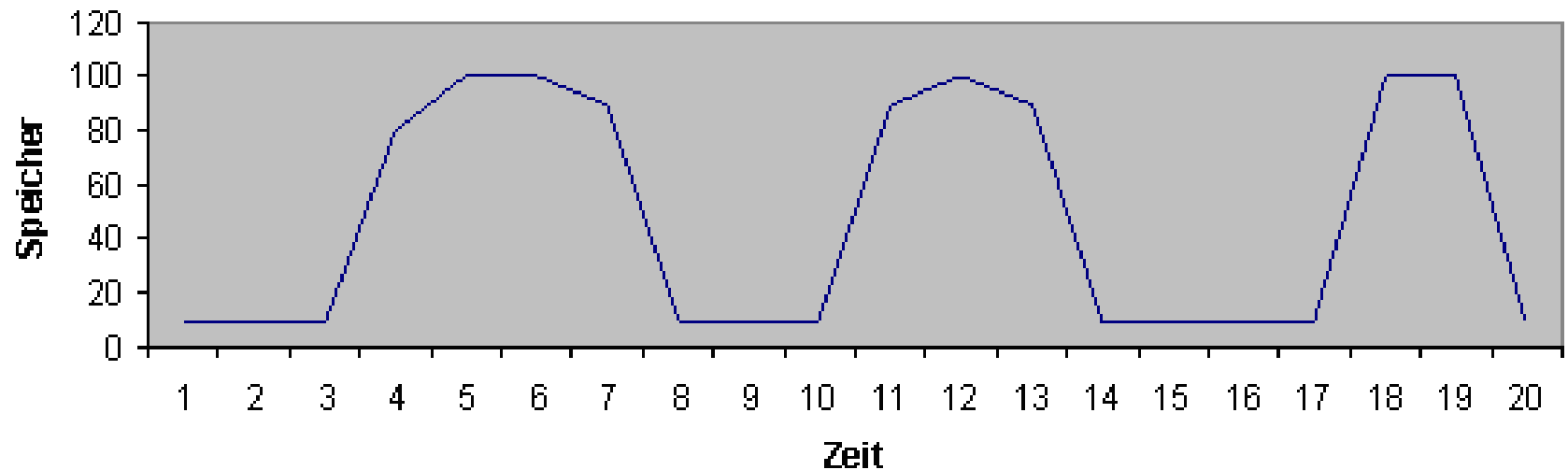
TYPISCHE BETRIEBSSITUATIONEN

- Die folgenden Abschnitte beschreiben eine Auswahl typischer Betriebssituationen einer lang laufenden virtuellen Maschine
- Diese Situationen sind natürlich idealisiert und stellen sich in der Realität wesentlich unklarer dar
 - Insbesondere zeigt sich bei allen lang laufenden Java-Anwendungen auch ohne Last ein langsames Anwachsen des Speicherbedarfs, der durch eine Full Garbage Collection bereinigt werden muss
- Nichtsdestotrotz können die hier beschriebenen Szenarien mit etwas Erfahrung erkannt und für die Analyse eines Problems sinnvoll eingesetzt werden

- Im Idealfall finden sich alle Requests, die vom Java-Server bearbeitet werden, als Peaks im Speicherverlauf wieder

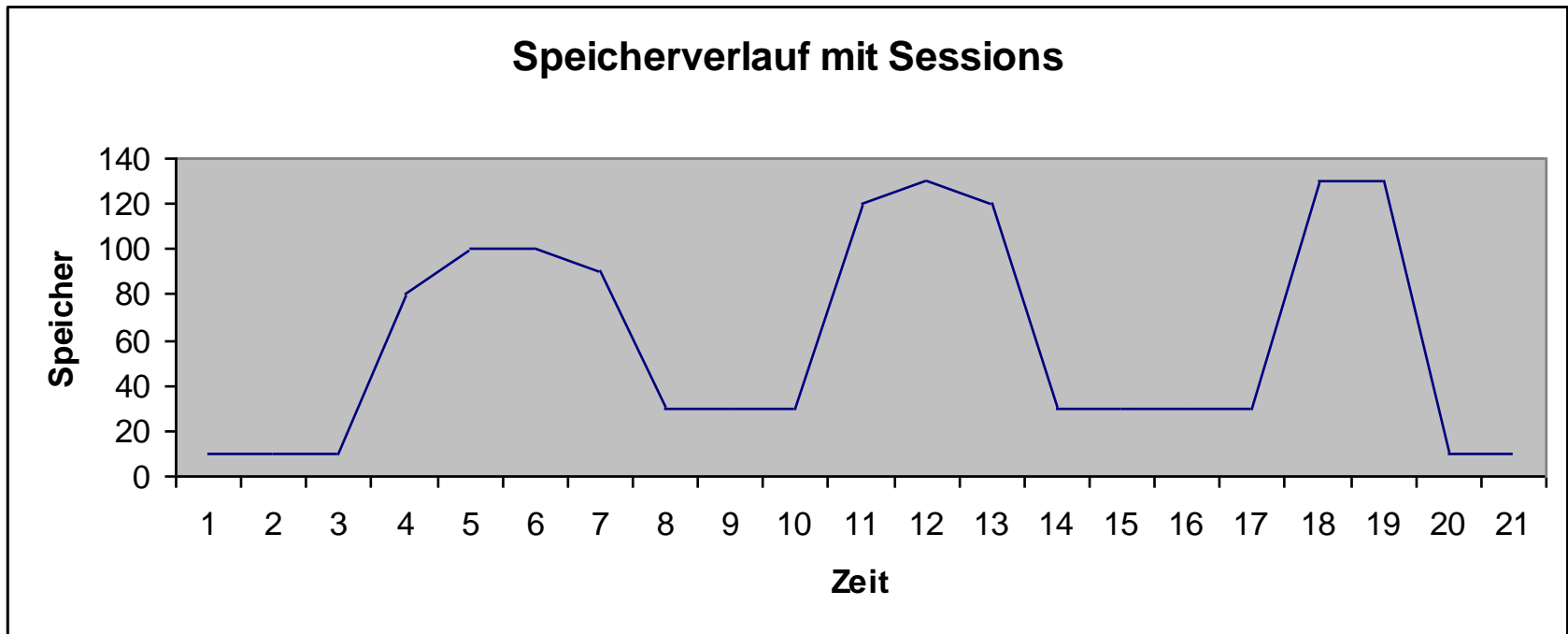


Idealer Speicherverlauf

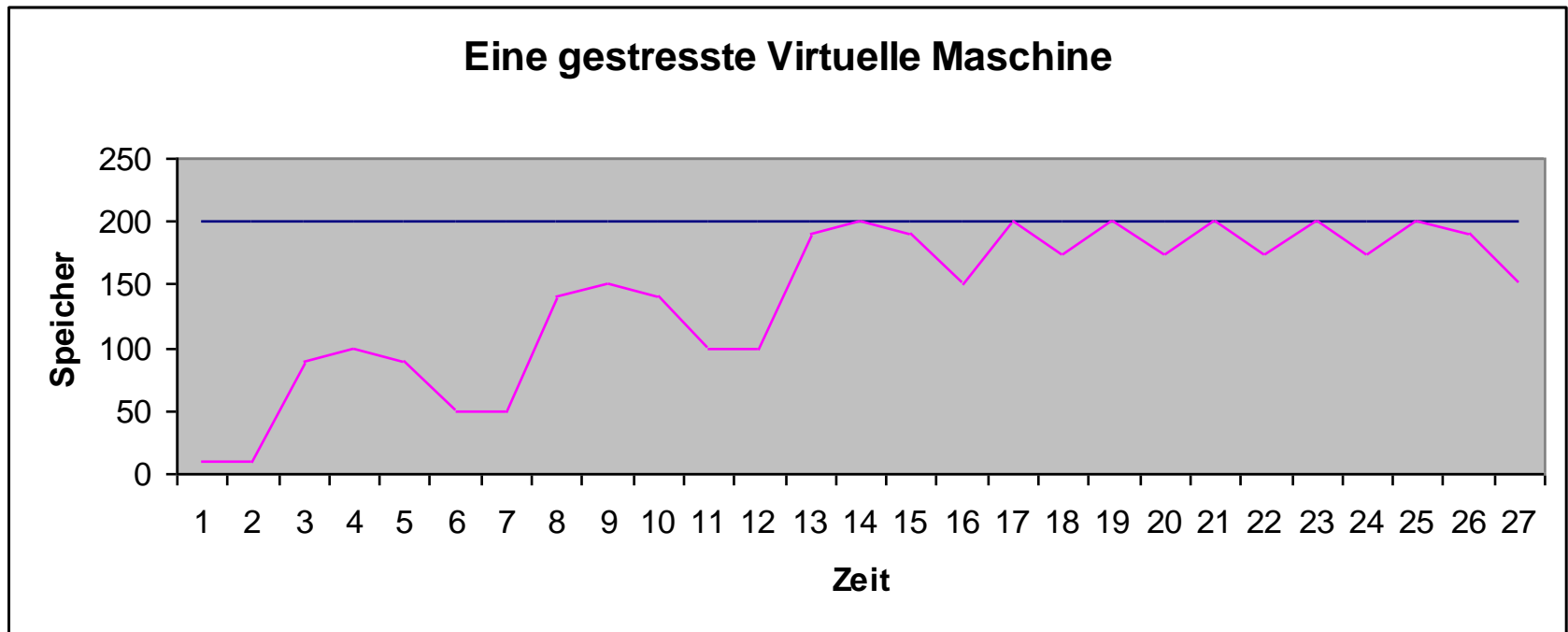


- Die Breite des Peaks hängt davon ab, wie lange die Request-Verarbeitung dauert
- Nach dem Ende des Requests wird hier exakt der ursprüngliche Wert des Speicherverbrauchs erreicht
 - Alle notwendigen Objekte können von der Garbage Collection entfernt werden
 - Bevorzugt sollten dies Copy-Collections innerhalb der New Generation sein

- Hier werden durch Requests auf dem Server zusätzlich Sessions (Login eines Anwenders) aufgebaut, die zu breiten Plateaus führen
 - Erst beim Beenden der Session (Logout) wird der Speicher wieder komplett bereinigt
- Durch die aus Sicht der Virtuellen Maschine lang laufenden Sessions wird hier mit Sicherheit die Tenured Generation genutzt werden

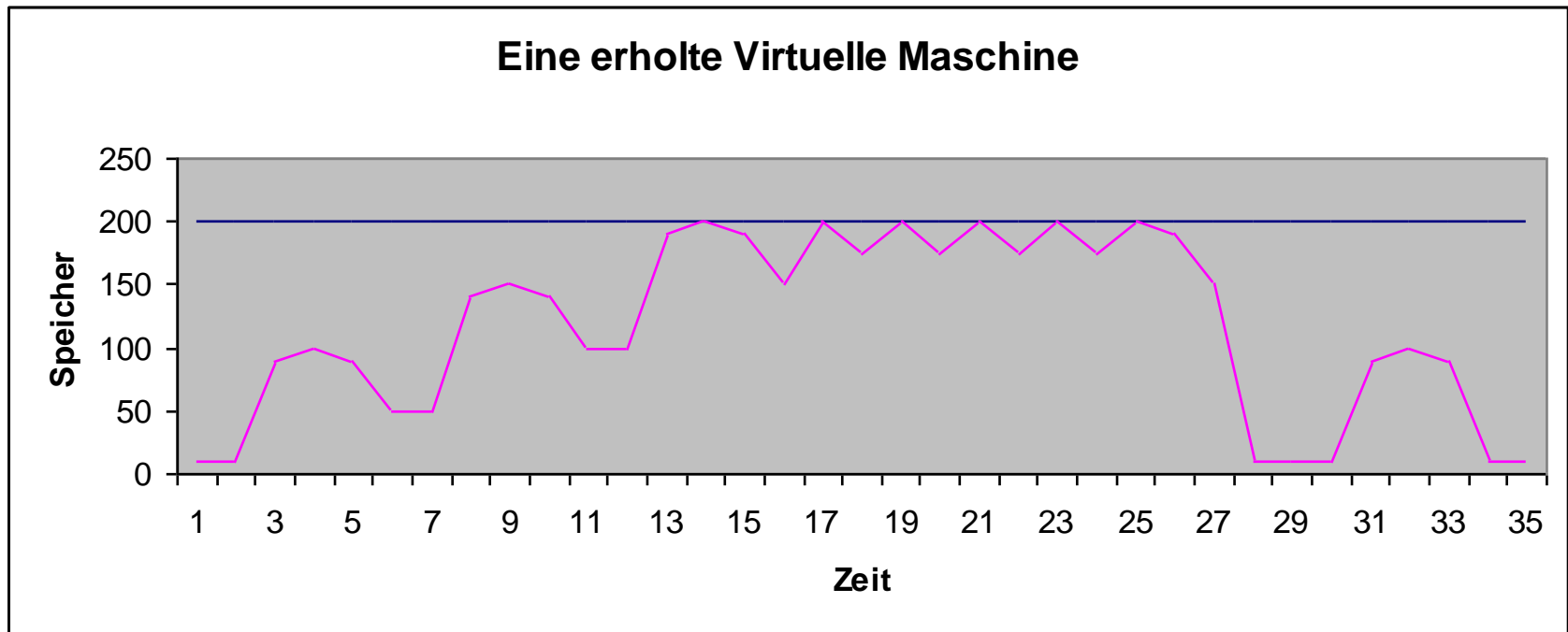


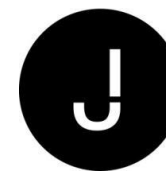
- Bei den bisher gezeigten Bildern hatte die Virtuelle Maschine kein Problem, den für die Request-Verarbeitung notwendigen Speicher zur Verfügung zu stellen
- Garbage Collections erfolgten jeweils nur am Ende der Request-Verarbeitung bzw. am Ende der Session
- Diese ändert sich jedoch sofort dann, wenn der Speicher die Maximalgrenze erreicht



- In diesem Beispiel wird in durch drei aufeinander folgende Requests eine Session von 150 Einheiten aufgebaut
- Die ersten drei Requests erfordern jeweils eine Garbage Collection
- Auf Grund der Speichernot löst der vierte Request nun aber eine ganze Kaskade von Garbage Collections aus, die zwangsläufig zu einer Verlängerung der Verarbeitungsdauer führt
- Aus Sicht des Anwenders dauert also der gleiche Vorgang länger als sonst

- Dieser Stress-Effekt ist reversibel
 - Wird die Session wieder abgebaut ist alles wieder in Ordnung
- Kann während der Stress-Phase für einen Request nicht mehr genügend Speicher angefordert werden, wird diese eine Verarbeitung mit einem `OutOfMemoryError` abgebrochen
 - Nach der Erholung ist aber alles wieder in Ordnung



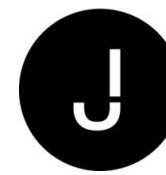


3.11

ALTERUNG DER JAVA VIRTUAL MACHINE

- Nicht immer kann der Speicher nach Abbau aller Sessions wieder den ursprünglichen Wert erreichen
- Baut die Anwendung beispielsweise einen permanenten Cache auf, den der Programmierer immer weiter befüllt, so wird damit ein Memory Leak erzeugt
 - Die Virtuelle Maschine bleibt dann im gestressten Zustand und kann sich nicht mehr erholen

- Dieser Zustand wird durch Überwachung leicht erkannt
- Die durchschnittliche Antwortzeit des Servers bzw. der Durchsatz sinkt im Laufe der Zeit kontinuierlich ab
- In den Log-Dateien tauchen immer häufiger Request-Abbrüche mit OutOfMemory-Situationen auf
- Eine gealterte Virtuelle Maschine kann sich nur durch Neustart „erholen“



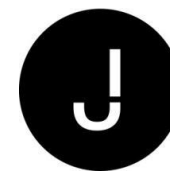
3.12

ANALYSE

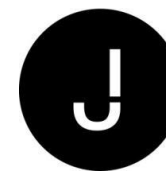
- Stress- und Alterungseffekte können nicht unbedingt durch eine Server-Konfiguration korrigiert werden
 - Natürlich kann das Auftreten der Probleme durch Einsatz von mehr Speicher zeitlich nach Hinten verschoben werden, aber das Problem tritt irgendwann dann doch auf
- Soll das Problem durch Änderung der Anwendung korrigiert werden ist es sinnvoll, den Entwicklern Informationen darüber zu geben, welche Objekte Speicher blockieren
 - Dies erfolgt durch einen Heap-Dump
 - Ein Administrator kann durch die Anwendung jmap jede Virtuelle Maschine zum Schreiben des Dumps veranlassen und diesen den Entwicklern zukommen lassen

- Dump-Dateien sind mindestens so groß wie der Speicher, der gedumped wird
 - Also auf Server-Umgebungen schnell bis zu mehreren Gigabytes
- Das Schreiben des Dumps bedingt zwangsläufig einen sehr ausgeprägten Stop-The-World-Effekt, der durchaus auch mehrere Minuten dauern kann
 - Das Auslösen eines Dumps im Produktionsbetrieb muss deshalb wohl überlegt und vorbereitet werden!

- Die Anwendung startet im interpretierten Modus und wird während ihrer Ausführung analysiert, um die sogenannten „Hot Spots“, also Programmteile, die sehr oft bzw. wiederholt ausgeführt werden, zu finden
- Ist ein Hotspot identifiziert, wird der originale Bytecode des Hotspots durch vom Compiler optimierten nativen Binärcode ersetzt
- Die Art und der Umfang der Optimierung sind dabei abhängig von der Version der Virtuellen Maschine und der verwendeten Variante (Client- oder Servervariante) des Hotspot-Compilers

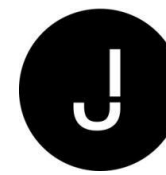


- Inlining
- Range Check Elimination
- Dead Code Elimination
- Loop Unrolling
- ...



4

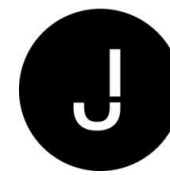
MANAGEMENT MIT JMX



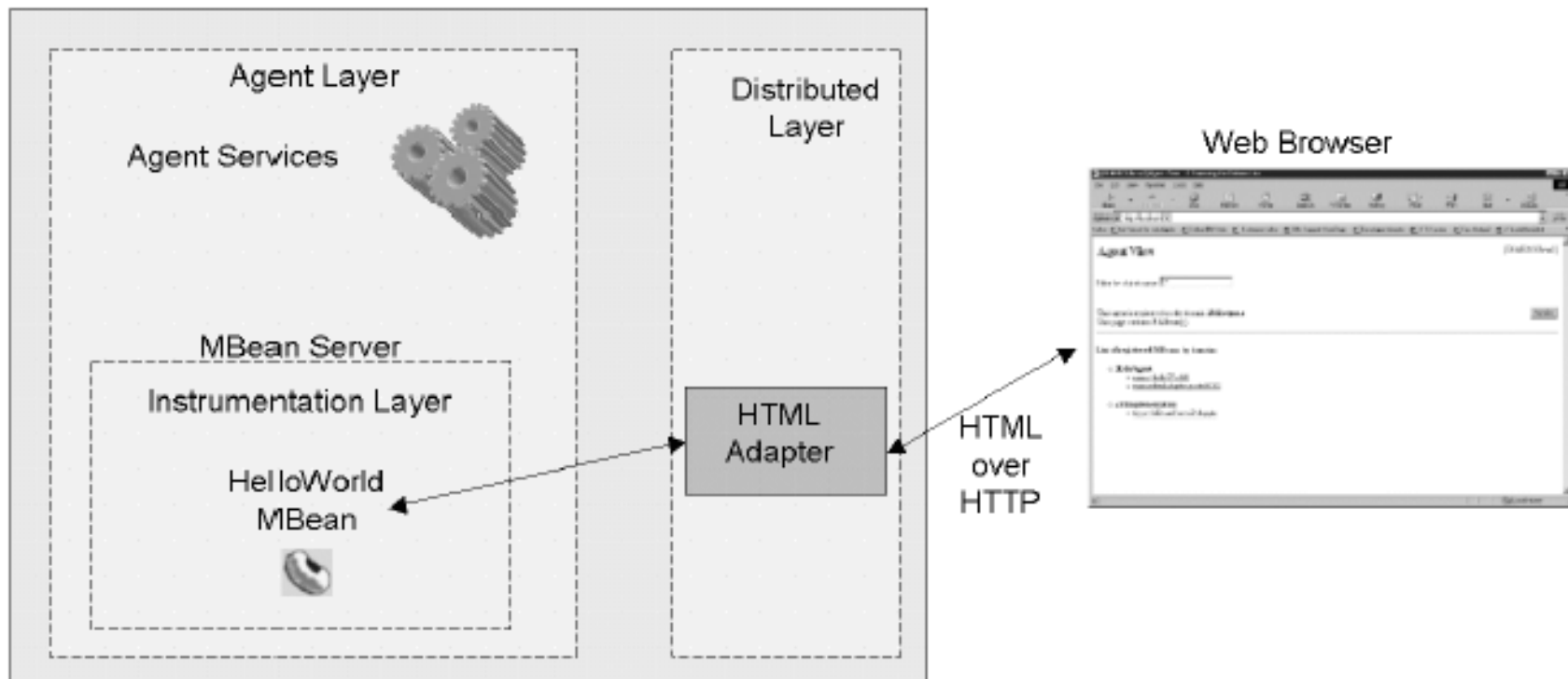
4.1

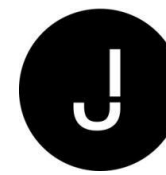
ÜBERBLICK

- Java Anwendungen sollen ohne großen Aufwand administriert werden können
 - Insbesondere solle die Lösung durch ein unabhängiges Modul realisiert werden, das vom Rest der Anwendung unabhängig ist
- Um dieses Ziel zu erreichen, definiert die Java Management Extension „JMX“ den JMX Server
- Dieser bietet ist ein Framework bzw. eine Laufzeitumgebung für Managed Beans:
 - Lebenszyklus,
 - Zustandsänderungen,
 - Events und Notification-Mechanismen



- Instrumentation Layer
- Agent Layer
- Distributed Services Layer
 - Innerhalb des Distributed Services Layer werden „normale“ Kommunikationsprotokolle verwendet:
 - http für Web basierte Administrationskonsolen
 - SNMP für vorhandene System
 - Java RMI
 - JINI
 - ...

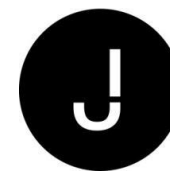




4.2

MBEANS

- Identifikation über einen eindeutigen Object Name
 - domain:key=value,key2=value2
- Jede Bean hat eine beliebige Menge von Eigenschaften/Attributen
 - Read-only
 - Read/Write
- Jede Bean hat Operationen
 - diese werden direkt im Server ausgeführt
- Der Notification-Mechanismus wird häufig nicht benutzt
 - Dies übernimmt eher ein zentrale Überwachungssoftware wie Nagios



- Spezielle JMX-Clients
 - jconsole als Bestandteil der Java-Installation
- Administrations- und Webkonsolen
 - Mehr oder weniger mächtige Werkzeuge, die die Hersteller von Applikationsservern zur Verfügung stellen
- Übergreifende Produkte
 - Jolokia
 - Eine simple Anwendung, die in jeden Applikationsserver installiert werden kann
 - Zugriff über simple http-Requests
 - cURL!
 - Integration in Überwachungssoftware damit fast trivial
 - HawtIO

[Home](#)[Download](#)[Features](#)[Documentation](#)[Support](#)[Blog](#)[About](#)

Jolokia

[Download](#)[Features](#)[Support](#)[Forum](#)[IRC](#)[License](#)

Documentation

[Overview](#)[Tutorial](#)[Reference Manual](#)[Talks and Screencasts](#)

Agents

[Overview](#)[Web Archive \(war\)](#)[Osgi](#)[JVM](#)[Mule](#)

Jolokia is remote JMX with JSON over HTTP.

It is fast, simple, polyglot and has unique features. It's JMX on Capsaicin.

Jolokia is a JMX-HTTP bridge giving an alternative to JSR-160 connectors. It is an agent based approach with support for many platforms. In addition to basic JMX operations it enhances JMX remoting with unique features like bulk requests and fine grained security policies.

Starting points

- Overview of [features](#) which make Jolokia unique for JMX remoting.
- The [documentation](#) includes a [tutorial](#) and a [reference manual](#).
- [Agents](#) exist for many platforms (JEE, OSGi, Mule, JVM).
- [Support](#) is available through various channels.
- [Contributions](#) are highly appreciated, too.

News

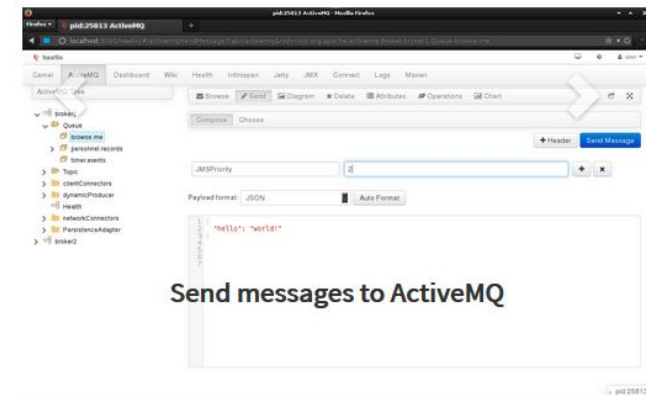


[Get Started Now!](#) [Documentation](#) [Community](#) [Demos](#) [github](#)

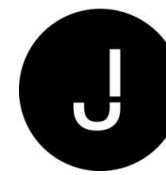
a **modular** web console for
managing your Java stuff

[View Demos](#)

[Get Started Now!](#)

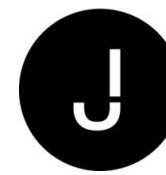


Send messages to ActiveMQ



5

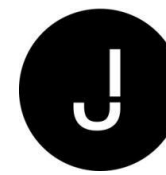
ENTERPRISE INTEGRATION PATTERNS



5.1

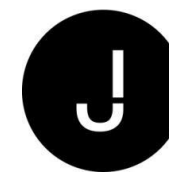
ÜBERSICHT




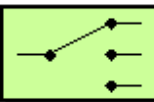
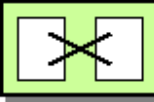
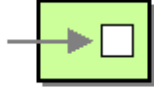
- Teile der Enterprise Integration Patterns sind bereits im Abschnitt über die Programmierung beschrieben worden
 - Auch ein Choice ist ein Pattern!
- In diesem Kapitel werden eher die System-relevanten Patterns beschrieben








5.2

KATALOG


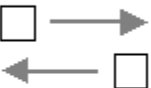
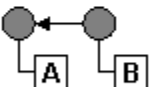



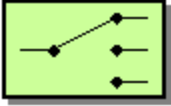

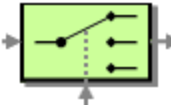
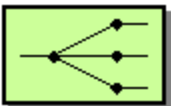
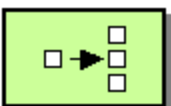

	Message Channel	How does one application communicate with another using messaging?
	Message	How can two applications connected by a message channel exchange a piece of information?
	Pipes and Filters	How can we perform complex processing on a message while maintaining independence and flexibility?
	Message Router	How can you decouple individual processing steps so that messages can be passed to different filters depending on a set of conditions?
	Message Translator	How can systems using different data formats communicate with each other using messaging?
	Message Endpoint	How does an application connect to a messaging channel to send and receive messages?

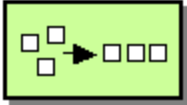
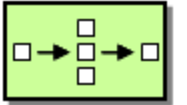
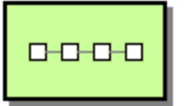
Messaging Channels

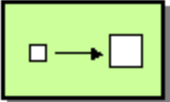
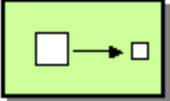
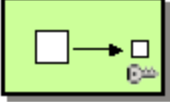

	Point to Point Channel	How can the caller be sure that exactly one receiver will receive the document or perform the call?
	Publish Subscribe Channel	How can the sender broadcast an event to all interested receivers?
	Dead Letter Channel	What will the messaging system do with a message it cannot deliver?
	Guaranteed Delivery	How can the sender make sure that a message will be delivered, even if the messaging system fails?
	Message Bus	What is an architecture that enables separate applications to work together, but in a de-coupled fashion such that applications can be easily added or removed without affecting the others?



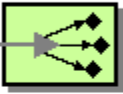

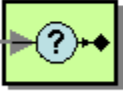
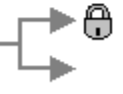
Message Construction

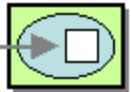
	Event Message	How can messaging be used to transmit events from one application to another?
	Request Reply	When an application sends a message, how can it get a response from the receiver?
	Correlation Identifier	How does a requestor that has received a reply know which request this is the reply for?
	Return Address	How does a replier know where to send the reply?

	Content Based Router	How do we handle a situation where the implementation of a single logical function (e.g., inventory check) is spread across multiple physical systems?
	Message Filter	How can a component avoid receiving uninteresting messages?
	Dynamic Router	How can you avoid the dependency of the router on all possible destinations while maintaining its efficiency?
	Recipient List	How do we route a message to a list of (static or dynamically) specified recipients?
	Splitter	How can we process a message if it contains multiple elements, each of which may have to be processed in a different way?
	Aggregator	How do we combine the results of individual, but related messages so that they can be processed as a whole?

	Resequencer	How can we get a stream of related but out-of-sequence messages back into the correct order?
	Composed Message Processor	How can you maintain the overall message flow when processing a message consisting of multiple elements, each of which may require different processing?
	Scatter-Gather	How do you maintain the overall message flow when a message needs to be sent to multiple recipients, each of which may send a reply?
	Routing Slip	How do we route a message consecutively through a series of processing steps when the sequence of steps is not known at design-time and may vary for each message?
	Throttler	How can I throttle messages to ensure that a specific endpoint does not get overloaded, or we don't exceed an agreed SLA with some external service?
	Sampling	How can I sample one message out of many in a given period to avoid downstream route does not get overloaded?
	Delayer	How can I delay the sending of a message?
	Load Balancer	How can I balance load across a number of endpoints?
	Multicast	How can I route a message to a number of endpoints at the same time?

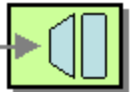
	Content Enricher	How do we communicate with another system if the message originator does not have all the required data items available?
	Content Filter	How do you simplify dealing with a large message, when you are interested only in a few data items?
	Claim Check	How can we reduce the data volume of message sent across the system without sacrificing information content?
	Normalizer	How do you process messages that are semantically equivalent, but arrive in a different format?
	Sort	How can I sort the body of a message?
	Script	How do I execute a script which may not change the message?
	Validate	How can I validate a message?

	Event Driven Consumer	How can an application automatically consume messages as they become available?
	Polling Consumer	How can an application consume a message when the application is ready?
	Competing Consumers	How can a messaging client process multiple messages concurrently?
	Message Dispatcher	How can multiple consumers on a single channel coordinate their message processing?
	Selective Consumer	How can a message consumer select which messages it wishes to receive?
	Durable Subscriber	How can a subscriber avoid missing messages while it's not listening for them?
	Idempotent Consumer	How can a message receiver deal with duplicate messages?



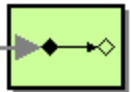
Transactional Client

How can a client control its transactions with the messaging system?



Messaging Gateway

How do you encapsulate access to the messaging system from the rest of the application?



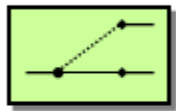
Service Activator

How can an application design a service to be invoked both via various messaging technologies and via non-messaging techniques?



ControlBus

How can we effectively administer a messaging system that is distributed across multiple platforms and a wide geographic area?



Detour

How can you route a message through intermediate steps to perform validation, testing or debugging functions?



Wire Tap

How do you inspect messages that travel on a point-to-point channel?

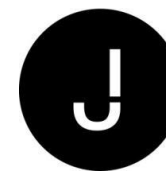
Message History

How can we effectively analyze and debug the flow of messages in a loosely coupled system?

Log

How can I log processing a message?

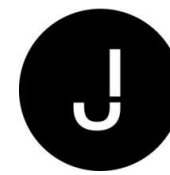
- Der Nachrichtenaustausch erfolgt an Hand von Austauschmustern
 - IN
 - OUT
 - INOUT
- Verschiedene Endpoints unterstützen mehrere Patterns
 - JMS mit oder ohne Listener zum Empfang einer Response-Message
- Für manche Endpoints kann das Standard-Verhalten geändert werden
 - One-way message für einen INOUT-Endpoint
 - One-way route mit Request-Response



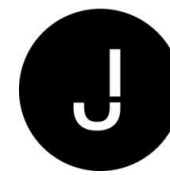
5.3

REALISIERUNG MIT APACHE CAMEL

- Content Based Routing
 - Steuerung des Routen-Ablaufs in Abhängigkeit vom Inhalt Exchange
- Filtering
- Wire Tap
 - Ein Exchange wird in eine zusätzliche Route gesendet
 - Sinnvoll beispielsweise für Logging/Auditing
- Multicast
 - Senden der Nachricht an mehrere Empfänger
- Recipient List
 - Nachrichtenversand an eine Liste von Empfängern
 - Dynamische Verwaltung der Liste möglich

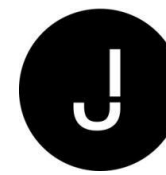


- **Throttler**
 - Beschränkt die Anzahl der Nachrichten, die an einen Endpoint gesendet werden
- **Dynamic Routing**
 - Route-Auswahl als Ergebnis eines Programmaufrufs
- **Load balancing**
 - über mehrere Endpoints



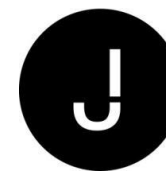
- Die Apache-Distribution enthält einen reichhaltigen Satz von Beispielprogrammen
 - Aggregation
 - Splitting
 - Parallelisierung

- **Transaktionelle Routen**
 - Endpoints können transaktionell sein
 - Die Nachrichten-Verarbeitung kann mit commit/rollback kontrolliert werden
- **Testen von Routen**
 - Die Camel-Distribution enthält JUnit-Erweiterungen
 - Ebenso existieren „Mock-Endpoints“
 - Inbound: Erzeugen von vordefinierten Messages
 - Outbound: Prüfen, ob und welche Nachrichten eingegangen sind



6

RESTFUL WEBSERVICES



6.1

DER REST-ANSATZ

- Doktorarbeit von Roy Fielding, 2000
- Siehe <http://en.wikipedia.org/wiki/REST>

Representational state transfer

From Wikipedia, the free encyclopedia
(Redirected from [REST](#))

"REST" redirects here. For other uses, see [Rest](#).

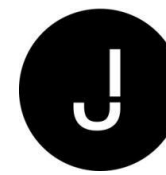
Representational state transfer (REST) is a software architectural style consisting of a coordinated set of architectural constraints applied to components, connectors, and data elements, within a distributed [hypermedia](#) system. REST ignores the details of component implementation and protocol syntax in order to focus on the roles of components, the constraints upon their interaction with other components, and their interpretation of significant data elements.^{[1][2]}

The term *representational state transfer* was introduced and defined in 2000 by [Roy Fielding](#) in his doctoral dissertation at [UC Irvine](#).^{[1][3]} REST has been applied to describe desired web architecture, to identify existing problems, to compare alternative solutions, and to ensure that protocol extensions would not violate the core constraints that make the web successful. Fielding used REST to design [HTTP 1.1](#) and [Uniform Resource Identifiers \(URI\)](#).^{[4][5]} The REST architectural style is also applied to the development of [web services](#)^[6] as an alternative to other distributed-computing specifications such as [SOAP](#).

Contents [\[hide\]](#)

- 1 History
- 2 Software architecture
 - 2.1 Components
 - 2.2 Connectors
 - 2.3 Data
- 3 Architectural properties
- 4 Architectural constraints
 - 4.1 Client-server
 - 4.2 Stateless
 - 4.3 Cacheable
 - 4.4 Layered system
 - 4.5 Code on demand (optional)
 - 4.6 Uniform interface

- Die Arbeitsweise des Internet abstrahiert
 - Was sind Ressourcen?
 - Wie werden Ressourcen identifiziert?
 - Was sind Ressourcen-Operationen?
 - Wie werden Services beschrieben?
 - Was sind Stateless Operationen?
 - Voraussetzungen und Umsetzung von Caching-Mechanismen
 - Optional: Übertragung von Skript-Logik auf den Client
- Grundlegendes Konzept sind die verlinkten HyperText-Dokumente
- Nicht überraschend: „Das Internet ist ein Beispiel für die Implementierung eines REST-basierten Systems“



6.2

REST UND HTTP

- Eine umfassende Spezifikation des w3w-Konsortiums
 - Siehe <http://en.wikipedia.org/wiki/Http>

Hypertext Transfer Protocol

From Wikipedia, the free encyclopedia
(Redirected from [Http](#))

The **Hypertext Transfer Protocol (HTTP)** is an [application protocol](#) for distributed, collaborative, [hypermedia](#) information systems.^[1] HTTP is the foundation of data communication for the [World Wide Web](#).

[Hypertext](#) is structured text that uses logical links ([hyperlinks](#)) between [nodes](#) containing text. HTTP is the protocol to exchange or transfer hypertext.

The standards development of HTTP was coordinated by the [Internet Engineering Task Force](#) (IETF) and the [World Wide Web Consortium](#) (W3C), culminating in the publication of a series of [Requests for Comments](#) (RFCs), most notably [RFC 2616](#) [\[4\]](#) (June 1999), which defines HTTP/1.1, the version of HTTP in common use.

Contents [\[hide\]](#)

- 1 Technical overview
- 2 History
- 3 HTTP session
- 4 Request methods
 - 4.1 Safe methods
 - 4.2 Idempotent methods and web applications
 - 4.3 Security
- 5 Status codes
- 6 Persistent connections
- 7 HTTP session state
- 8 Encrypted connections
- 9 Request message
- 10 Response message
- 11 Example session
 - 11.1 Client request

Internet protocol suite

Application layer

BGP • DHCP (DHCPv6) • DNS • FTP •
HTTP • IMAP • IRC • LDAP • MGCP •
NNTP • NTP • POP • RPC • RTP • RTSP •
RIP • SIP • SMTP • SNMP • SOCKS • SSH •
Telnet • TLS/SSL • XMPP • *more...*

Transport layer

TCP • UDP • DCCP • SCTP • RSVP •
more...

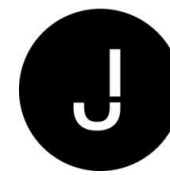
Internet layer

IP (IPv4 • IPv6) • ICMP • ICMPv6 • ECN •
IGMP • IPsec • *more...*

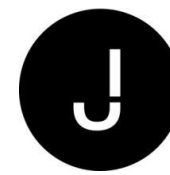
Link layer

ARP/InARP • NDP • OSPF • Tunnels (L2TP)
• PPP • Media access control (Ethernet •
DSL • ISDN • FDDI • DOCSIS) • *more...*

V • T • E



- Definition von URIs
 - Pfad
 - Parameter
- http-Request und http-Response
 - Daten-Container mit Header und Body
 - Encodierung
- Umfassender Satz von Header-Properties
 - Content-Length
 - Accepts
 - Content-Type



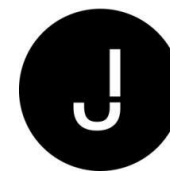
- http-Methoden
 - PUT
 - GET
 - POST
 - DELETE
 - OPTIONS
 - HEAD
- Statuscodes für Aufrufe
 - 404: „Not found“
 - 204: „Created“
 - ...

- Definition der Datentypen des Internet
 - Nicht zu verwechseln mit einem XML-Schema
 - Ein MimeType ist „nur“ eine strukturierte Zeichenkette
 - Eigene Erweiterungen sind möglich

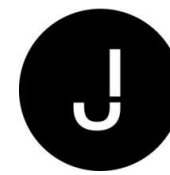
- REST hat mit http prinzipiell nichts zu tun
 - REST ist eine abstrakte Architektur
 - http ist ein konkretes Kommunikationsprotokoll
- Aber
 - http passt als Kommunikations-Protokoll der „Referenz-Implementierung“ Internet natürlich perfekt zum REST-Stil

- http Methoden und Ressourcen-Operationen
 - PUT
 - Neu-Anlegen einer Ressource
 - Aktualisierung
 - GET
 - Lesen einer Ressource
 - POST
 - Aktualisierung
 - Neuanlage
 - DELETE
 - Löschen
- Idempotenz
 - Idempotente Operationen dürfen beliebig oft aufgerufen werden und verursachen keine Nebeneffekte
 - REST verlangt eine idempotente Implementierung für PUT und DELETE

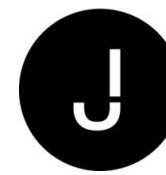
- Mit PUT
 - Der Client muss die Ressourcen-ID mit angeben
 - Rückgabe ist ein Statuscode „201: Created“
- Mit POST
 - Der Server entscheidet, ob er eine neue Ressource anlegen muss
 - Falls ja:
 - Statuscode „201: Created“
 - Gesetzter `Location`-Header mit URI der eben angelegten Ressource
 - Optional: Body enthält die angelegte Ressource



- Mit PUT
 - Statuscode „200: OK“ oder „204: No content“
 - PUT ist idempotent (!)
- Mit POST
 - POST wird für nicht-idempotente Updates benutzt

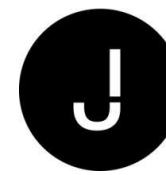


- Mit DELETE
 - Statuscode „200: OK“ oder „204: No content
 - PUT ist idempotent (!)
- Konzeptionell muss unterschieden werden:
 - Ein „echtes“ DELETE löscht die Ressource
 - Ein fachliches Löschen (z.B. Storno) ist eigentlich ein Update der Ressource
 - Ein überladen des http-DELETE ist für diese Zwecke jedoch durchaus legitim
 - `DELETE order/ISBN42?cancel=true`



7

BEISPIELE



7.1

ÜBERSICHT

- Bestandteil der ServiceMix-Distribution
 - Verzeichnis `examples`
- Realisierung als Maven-Projekt
 - Zur Analyse damit am einfachsten mit Eclipse/STS zu öffnen

 **activemq**

 **activiti**

 **akka**

 **camel**

 **cxfr**

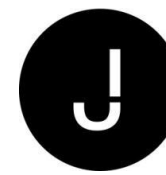
 **drools**

 **karaf**

 **META-INF**























 **pom.xml**

 **README.txt**



7.2

AUSFÜHRUNG

- ▶  activemq
- ▶  activiti
- ▶  akka
- ▼  camel
 - ▼  camel-blueprint
 - ▶  src
 -  org.apache.servicemix.examples.cfg
 -  pom.xml
 -  README.txt
 - ▶  camel-cxf-rest
 - ▶  camel-cxf-soap
 - ▶  camel-drools
 - ▶  camel-drools-blueprint
 - ▶  camel-osgi
 - ▶  camel-sql
 -  pom.xml
 - ▶  cxf
 - ▶  drools
 - ▶  karaf
 - ▶  META-INF
 -  pom.xml
 -  README.txt

- Bauen der Anwendungen durch Maven
 - mvn install
- Kopieren der generierten Artefakte in das Deployment-Verzeichnis von ServiceMix