Systems Integration in the NoSQL Era with Apache Camel

Kai Wähner

kwaehner@talend.com

@KaiWaehner

www.kai-waehner.de

4/18/2013

Kai Wähner

Consulting
Developing
Coaching
Speaking
Writing

Main Tasks

Requirements Engineering

Enterprise Architecture Management

Business Process Management

Architecture and Development of Applications

Service-oriented Architecture

Integration of Legacy Applications

Cloud Computing

Big Data

Contact

Email: kwaehner@talend.com

Blog: www.kai-waehner.de/blog

Twitter: @KaiWaehner

Social Networks: Xing, LinkedIn

talend* integration at any scale

What is the problem?

Growth

- Applications
- Interfaces
- Technologies
- Products

A new era: NoSQL

Solution: systems integration

All Roads lead to Rome ...

talend* integration at any scale

Wishes for integrators

- Standardized Modeling
- Efficient Realization
- Automatic Testing

Systems integration in the NoSQL era

What is the key message?

Key messages

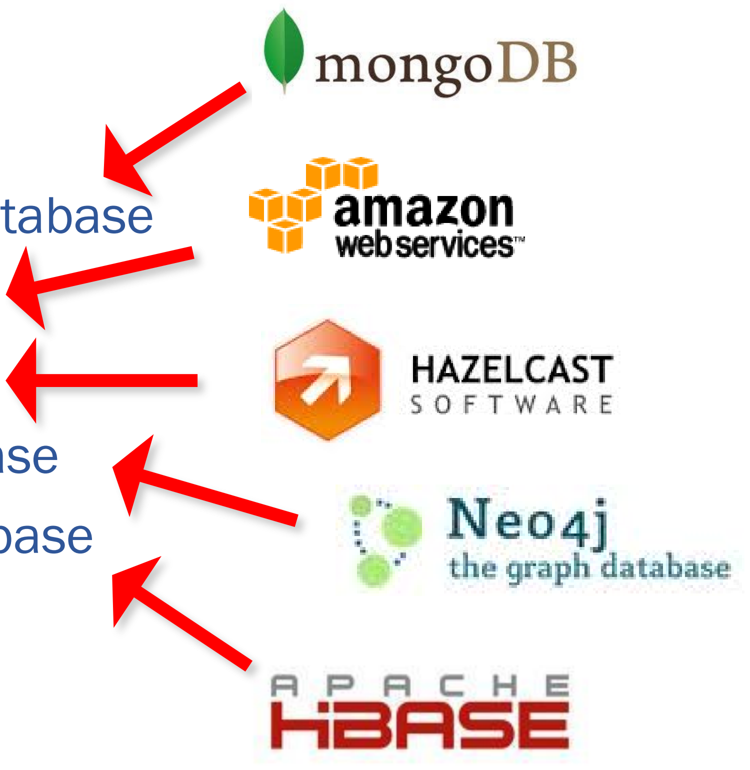
NoSQL cannot be avoided, and must be integrated!

NoSQL integration is already possible!

Apache Camel helps a lot!

Agenda

1) Introduction to NoSQL
2) Introduction to Apache Camel
3) Integration of a Document-oriented Database
4) Integration of a Key-Value Database
5) Integration of an In-Memory Database
6) Integration of a Graph-oriented Database
7) Integration of a Column-oriented Database
8) Custom NoSQL Components



Live Demos

Agenda

1) Introduction to NoSQL

2) Introduction to Apache Camel

3) Integration of a Document-oriented Database

4) Integration of a Key-Value Database

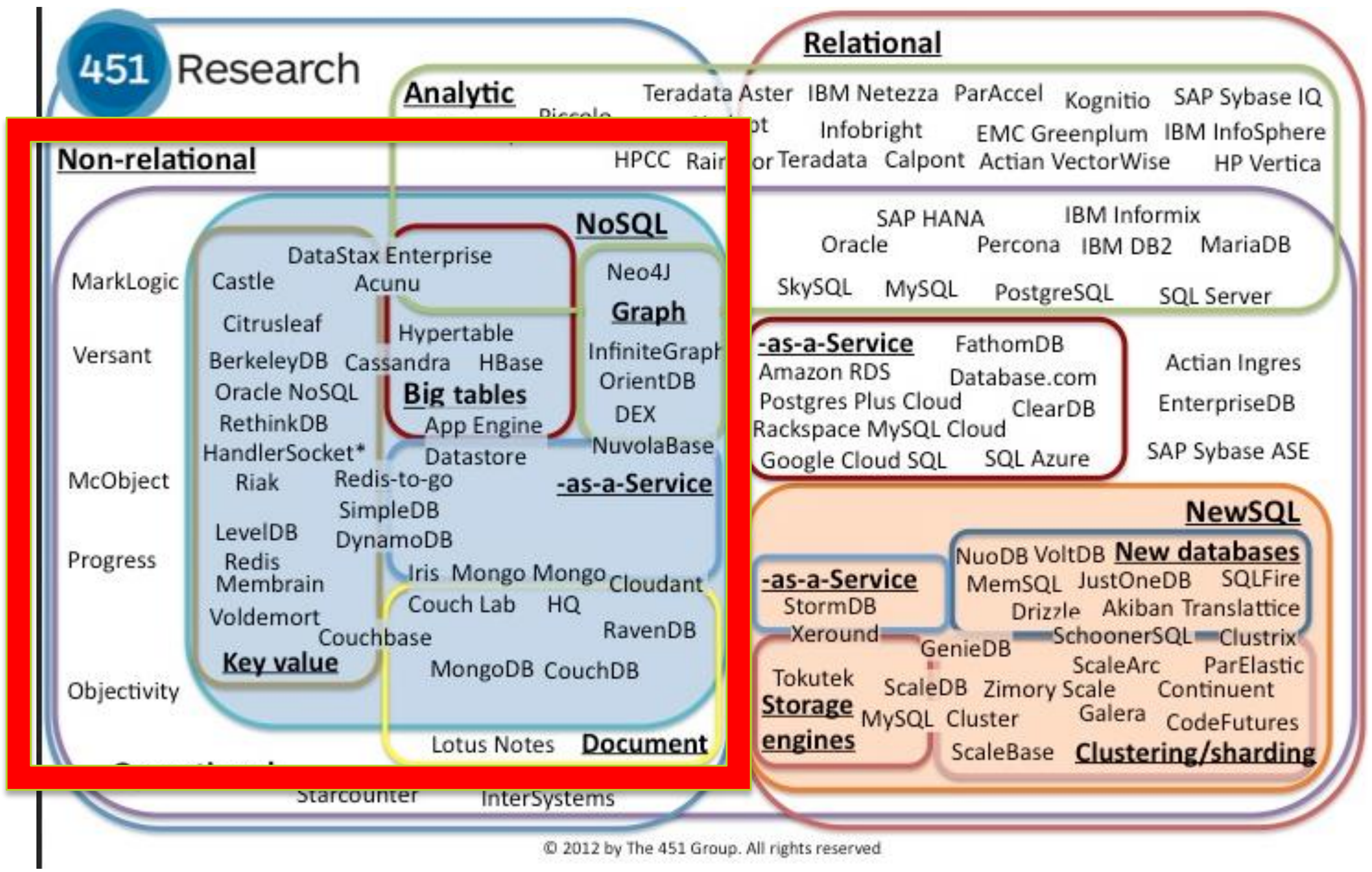
5) Integration of an In-Memory Database

6) Integration of a Graph-oriented Database

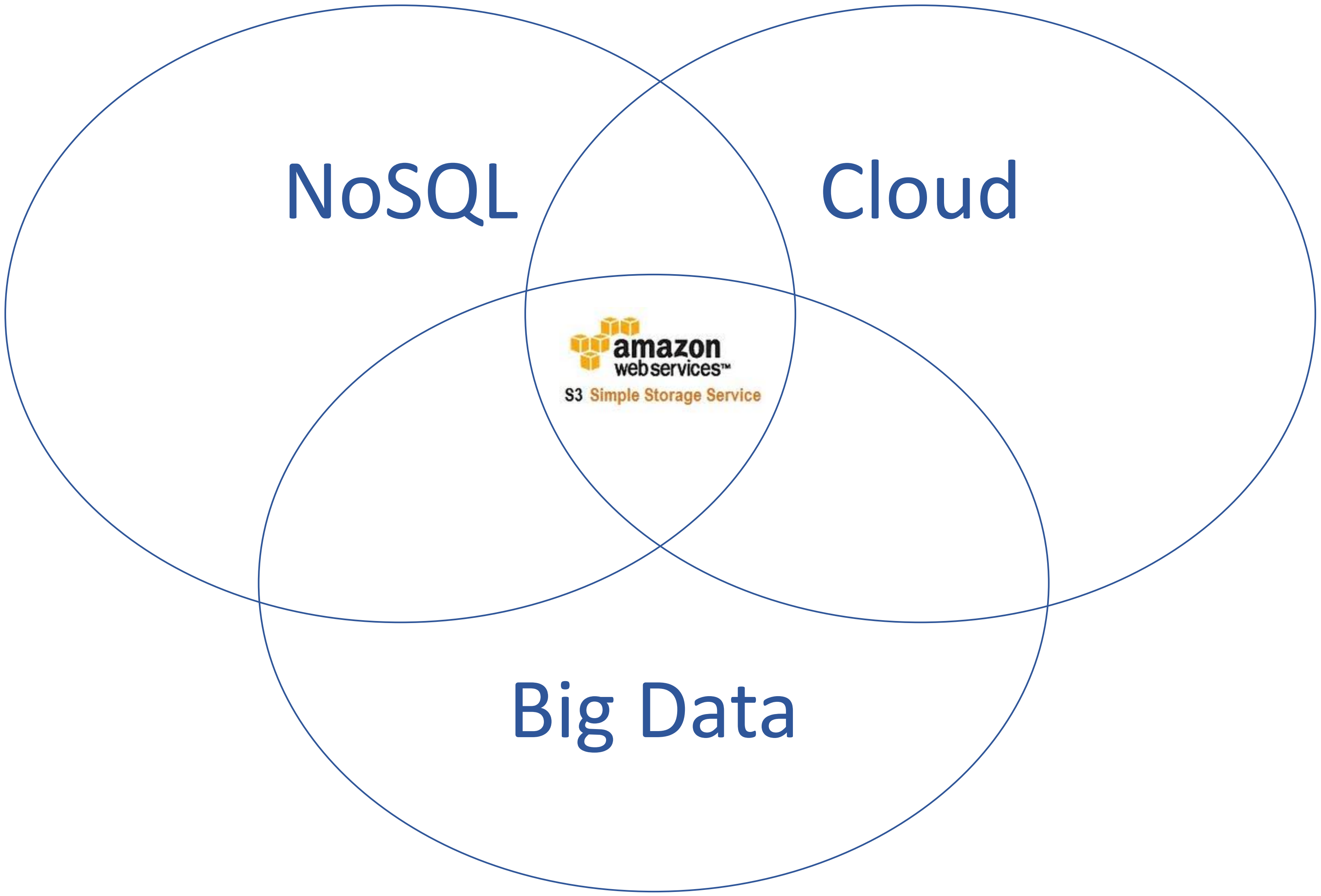
7) Integration of a Column-oriented Database

8) Custom NoSQL Components

The evolving database landscape



Complementary Concepts

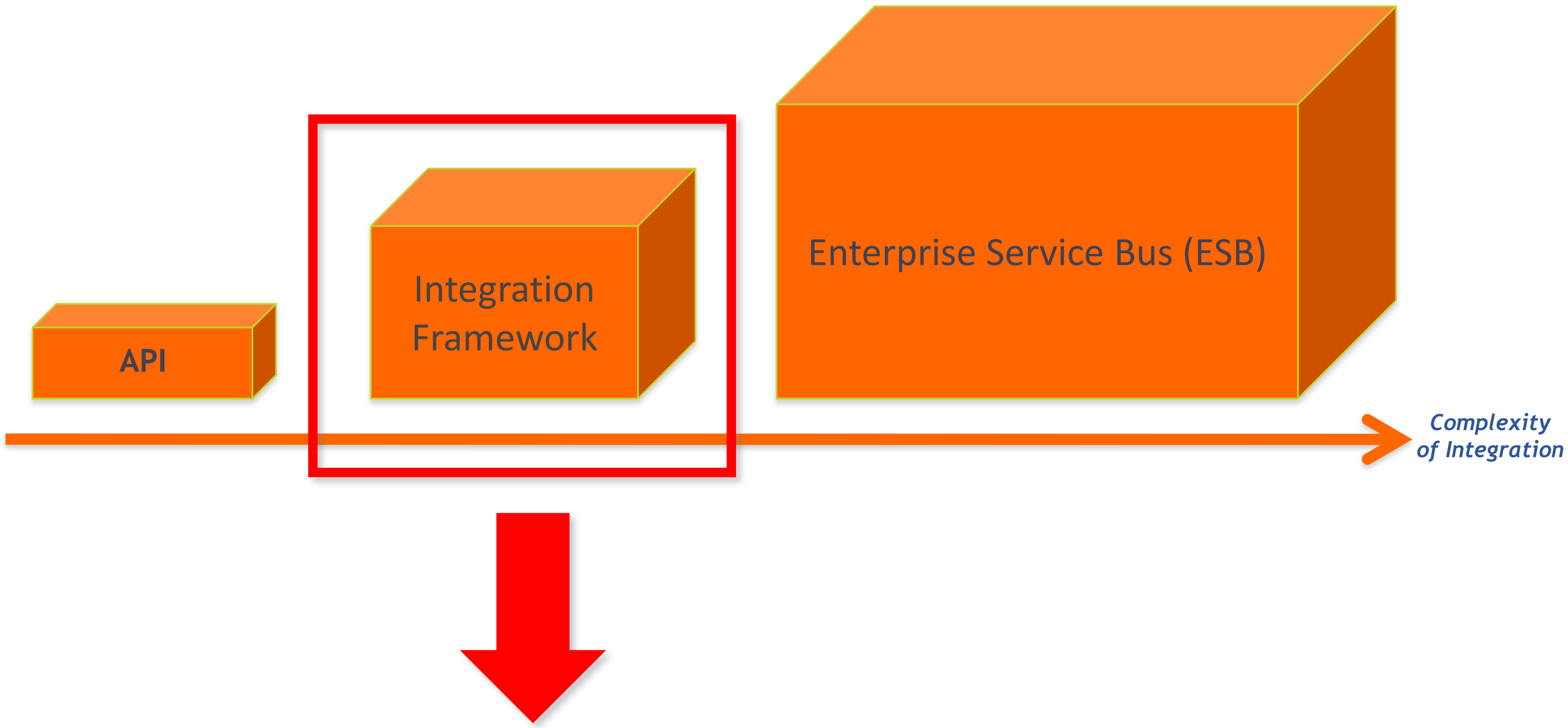


Agenda

1) Introduction to NoSQL
2) Introduction to Apache Camel
3) Integration of a Document-oriented Database
4) Integration of a Key-Value Database
5) Integration of an In-Memory Database
6) Integration of a Graph-oriented Database
7) Integration of a Column-oriented Database
8) Custom NoSQL Components

talend* integration at any scale

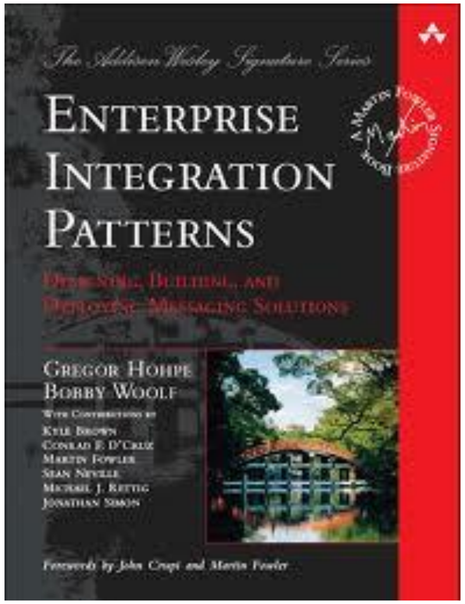
Alternatives for systems integration



Apache Camel vs. Spring Integration vs. Mule

http://www.kai-waehner.de/blog/2012/01/10/spoilt-for-choice-which-integration-framework-to-use-spring-integration-mule-esb-or-apache-camel/

Enterprise Integration Patterns (EIP)


The Addison-Wesley Signature Series

ENTERPRISE INTEGRATION PATTERNS

DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS

A MARTIN FOWLER SIGNATURE BOOK

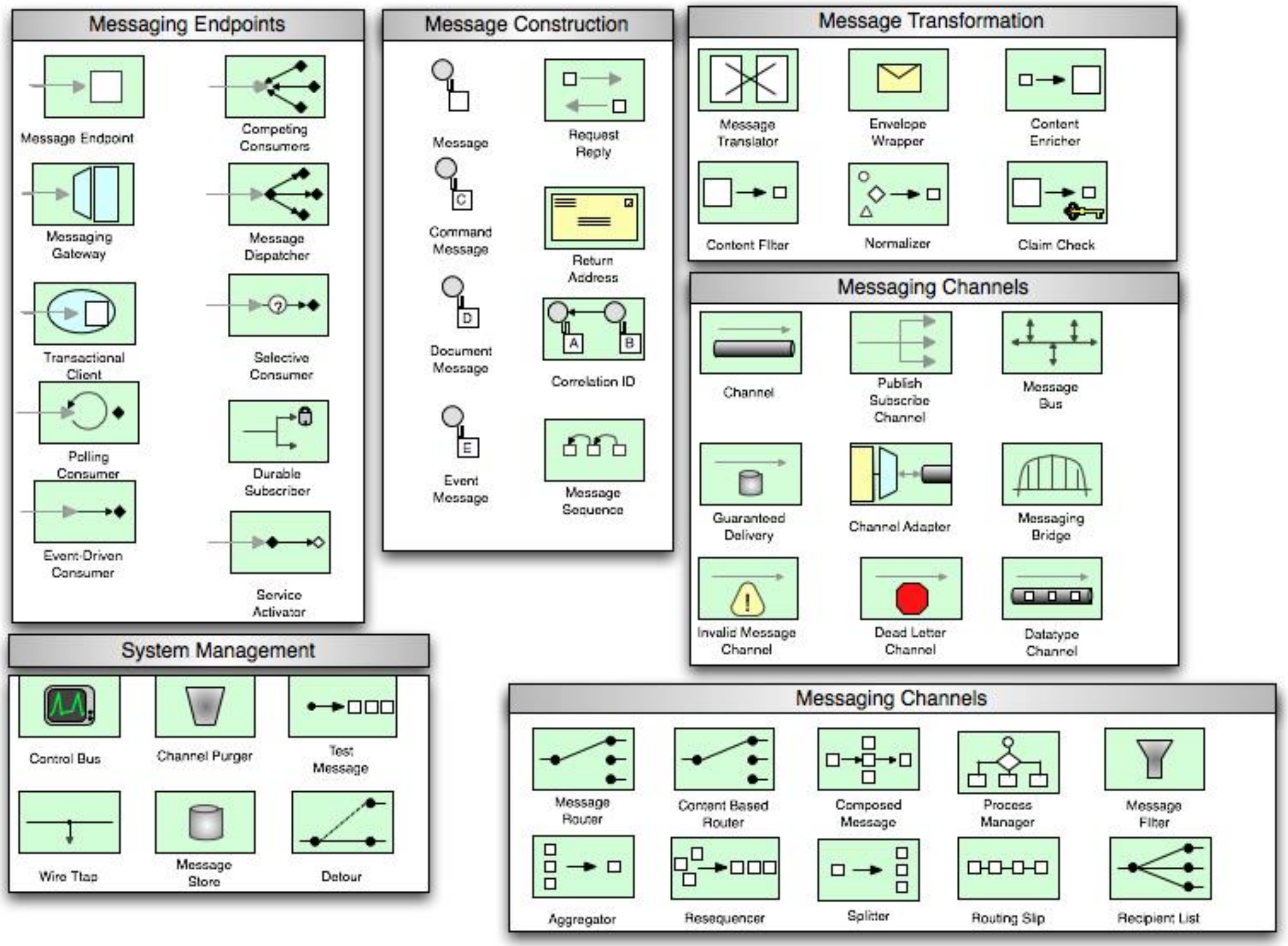
GREGOR HOHPE
BOBBY WOOLF

WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F. D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON

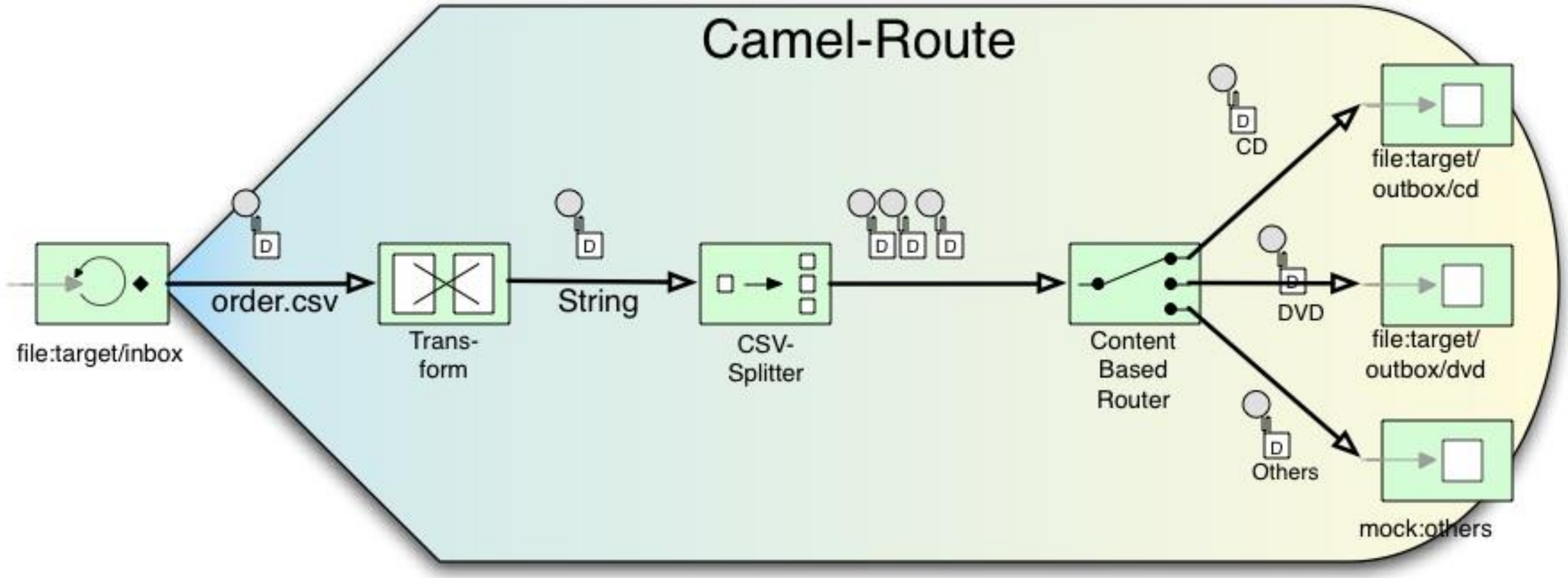
Forewords by John Crupi and Martin Fowler

Apache Camel
Implements the EIPs

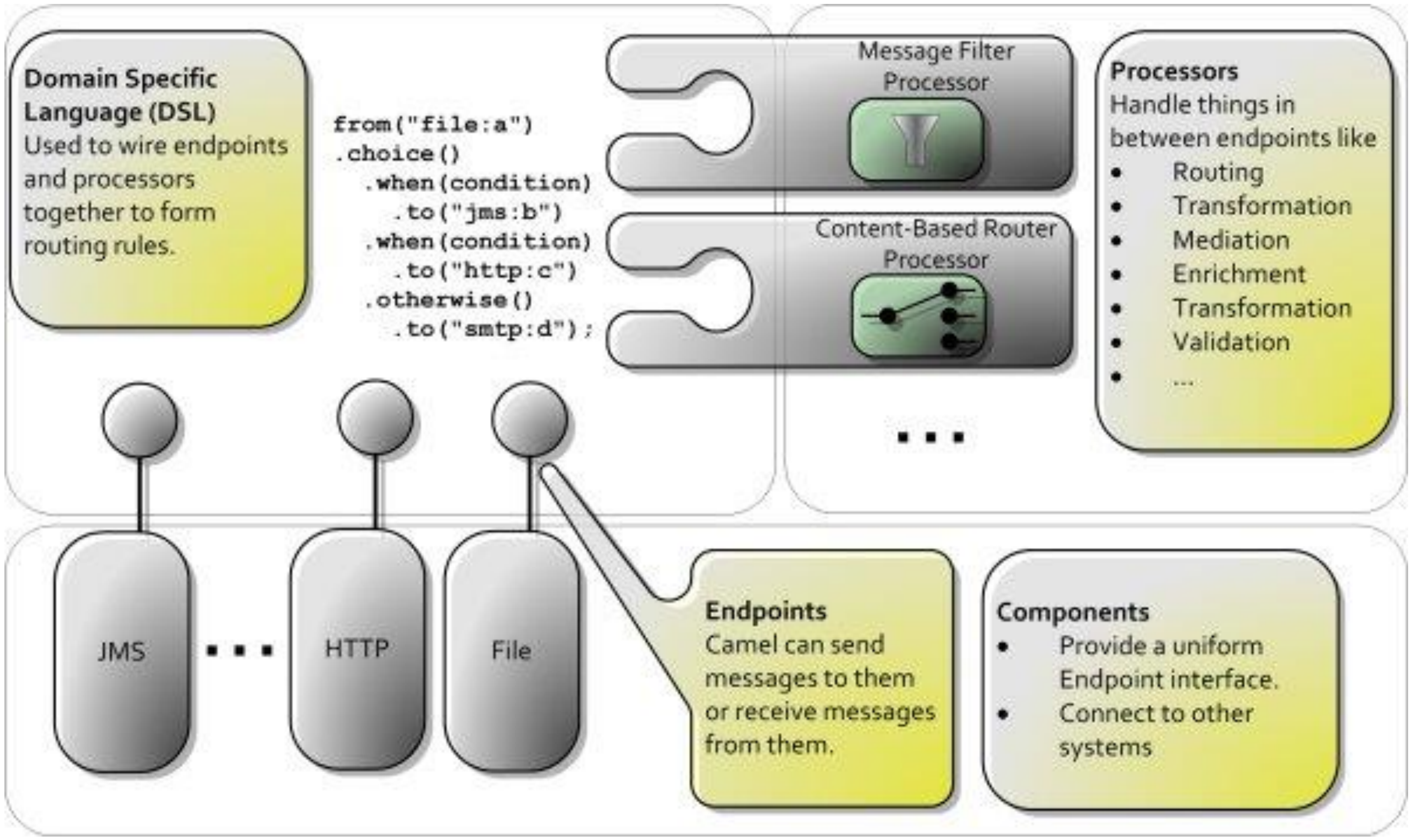
Enterprise Integration Patterns (EIP)



Enterprise Integration Patterns (EIP)



Architecture


Domain Specific Language (DSL)
Used to wire endpoints and processors together to form routing rules.
Message Filter Processor
Content-Based Router Processor
Processors
Handle things in between endpoints like
• Routing
• Transformation
• Mediation
• Enrichment
• Transformation
• Validation
• ...
JMS
HTTP
File
Endpoints
Camel can send messages to them or receive messages from them.
Components
• Provide a uniform Endpoint interface.
• Connect to other systems

```
from("file:a")
.choice()
   .when(condition)
     .to("jms:b")
   .when(condition)
     .to("http:c")
   .otherwise()
     .to("smtp:d");
```

http://java.dzone.com/articles/apache-camel-integration

Choose your favorite DSL

Java

XML

Scala

Groovy

Kotlin
by JetBrains
(not production-ready yet)

Choose your required components

TCP

SQL SMTP JMS

Netty Jetty

RMI

FTP Lucene JDBC EJB

Bean-Validation MQ IRC

JMX

Quartz

RSS AMQP

Atom Log

AWS-S3 HTTP

XSLT

LDAP

File

Akka

Many many more CXF Custom Components

Deploy it wherever you need

Standalone

Application Server

Web Container

Spring Container

OSGi

Cloud

talend* integration at any scale

Enterprise-ready

- Open Source
- Scalability
- Error Handling
- Transaction
- Monitoring
- Tooling
- Commercial Support

Community → Camel rocks!

Mailing Lists?

Forums?

Blogs?

Articles?

Conference talks?

ESBs?

Professionals?

Jobs?

Knowledge?

Live demo

Apache Camel in action…

Agenda

1) Introduction to NoSQL
2) Introduction to Apache Camel
3) Integration of a Document-oriented Database
4) Integration of a Key-Value Database
5) Integration of an In-Memory Database
6) Integration of a Graph-oriented Database
7) Integration of a Column-oriented Database
8) Custom NoSQL Components

Document-oriented database

mongoDB

OrientDB®

ArangoDB

CouchDB relax

sedna

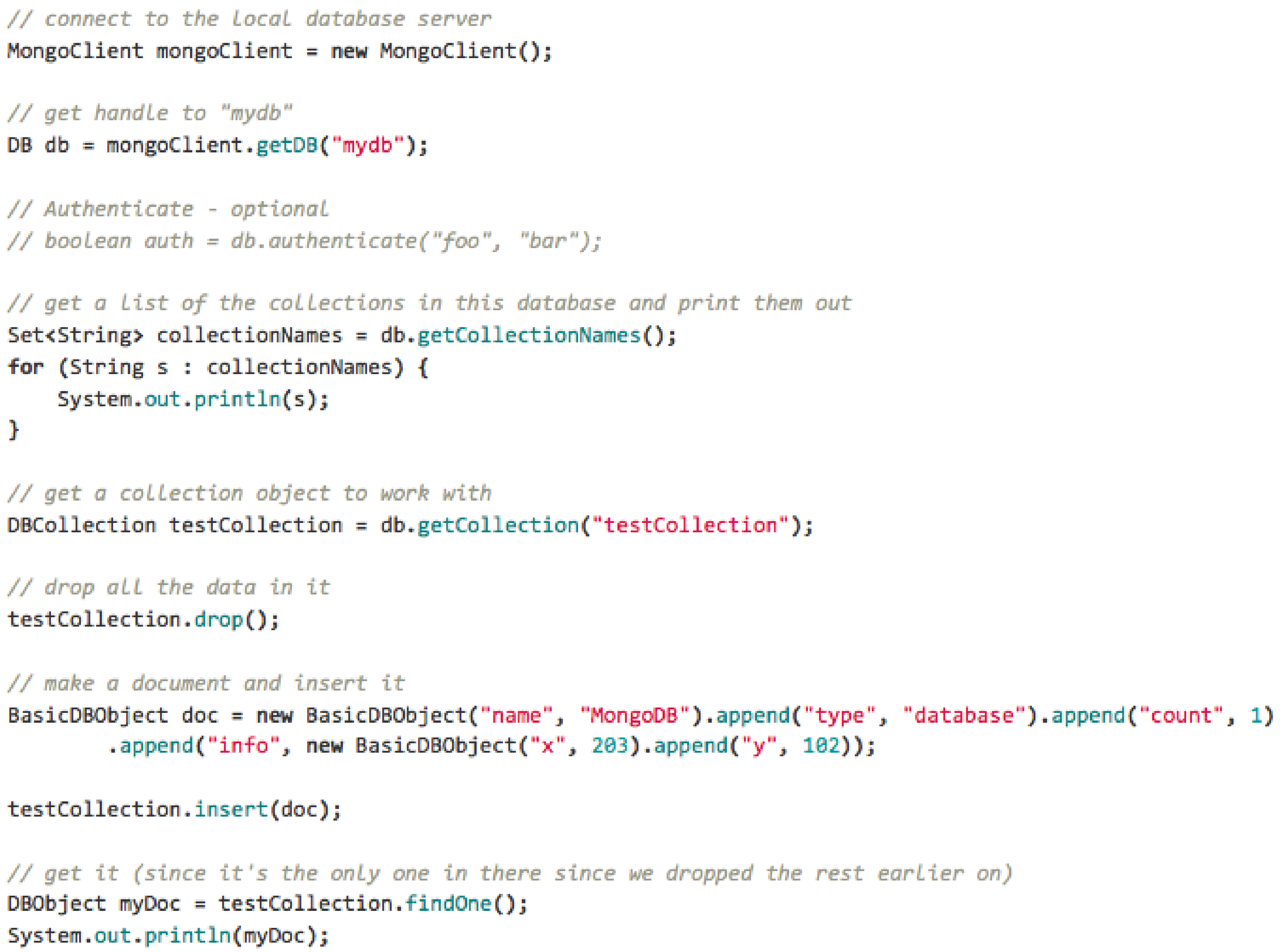
Native XML Database System

Document-oriented database

mongoDB

- 10gen
- stores structured data as JSON-like documents with dynamic schemas
- REST API and several SDKs (Java, .NET, Ruby, PHP, Python, etc.)
- Ad hoc queries, indexing, replication, load balancing
- Powerful, but also easy to use and flexible
- Example: Disney persists state information of online games in a common object repository.

Code example: MongoDB Java Driver



```java
// connect to the local database server
MongoClient mongoClient = new MongoClient();

// get handle to "mydb"
DB db = mongoClient.getDB("mydb");

// Authenticate - optional
// boolean auth = db.authenticate("foo", "bar");

// get a list of the collections in this database and print them out
Set<String> collectionNames = db.getCollectionNames();
for (String s : collectionNames) {
    System.out.println(s);
}

// get a collection object to work with
DBCollection testCollection = db.getCollection("testCollection");

// drop all the data in it
testCollection.drop();

// make a document and insert it
BasicDBObject doc = new BasicDBObject("name", "MongoDB").append("type", "database").append("count", 1)
        .append("info", new BasicDBObject("x", 203).append("y", 102));

testCollection.insert(doc);

// get it (since it's the only one in there since we dropped the rest earlier on)
DBObject myDoc = testCollection.findOne();
System.out.println(myDoc);
```

Code example: camel-mongodb component

```
// Producer
from(“jms:FlightDocumentQueue")
     .to("mongodb:myDb?database=flights
                        &collection=tickets
                        &operation=insert");

// Consumer
from("mongodb:myDb?database=flights
                        &collection=cancellations
                        &tailTrackIncreasingField=departureTime")
     .to(“jms:CancelledFlightsQueue");
```

Live demo

mongoDB

Integration of a document-oriented database in action…

Agenda

1) Introduction to NoSQL
2) Introduction to Apache Camel
3) Integration of a Document-oriented Database
4) Integration of a Key-Value Database
5) Integration of an In-Memory Database
6) Integration of a Graph-oriented Database
7) Integration of a Column-oriented Database
8) Custom NoSQL Components

Key-Value database

amazon webservices™
S3 Simple Storage Service

riak

rackspace

amazon webservices
Amazon DynamoDB

Key-Value database

amazon webservices™

S3 Simple Storage Service

- Part of Amazon Web Services (AWS)
- Online storage web service
- Store arbitrary objects (computer files) up to 5 terabytes
- REST and SOAP API
- SDKs for Java, .NET, PHP, Ruby, etc.
- Highly-scalable, reliable, and low-latency
- Alternative for Hadoop's file system HDFS
- Example: DigitalChalk offers creating, delivering and managing training videos

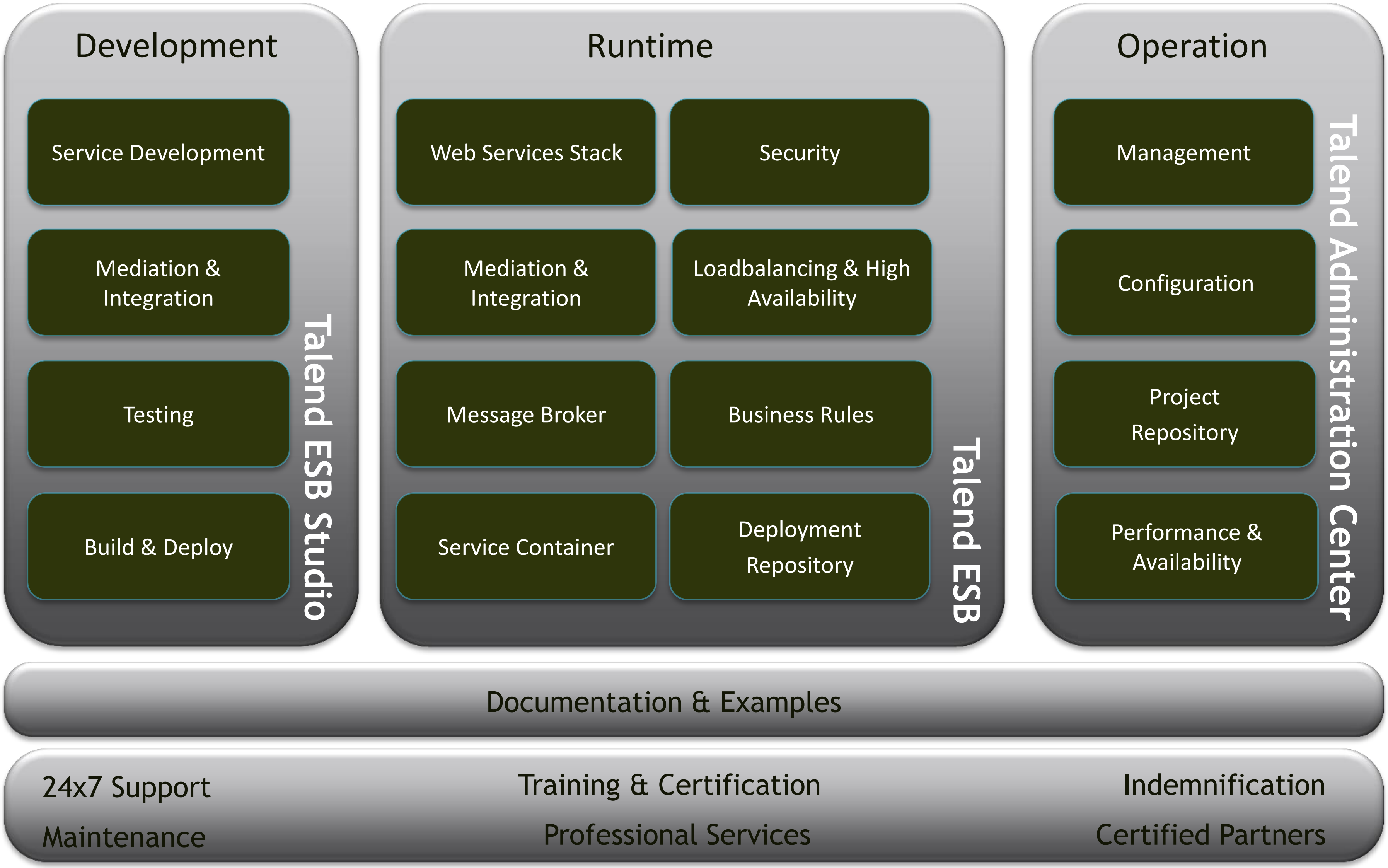
Code example: AWS S3 Java SDK

```
AmazonS3 s3 = new AmazonS3Client(new PropertiesCredentials(
        S3Sample.class.getResourceAsStream("AwsCredentials.properties")));

String bucketName = "my-first-s3-bucket-" + UUID.randomUUID();
String key = "MyObjectKey";

try {

    s3.createBucket(bucketName);
    s3.putObject(new PutObjectRequest(bucketName, key, createSampleFile()));

    S3Object object = s3.getObject(new GetObjectRequest(bucketName, key));

    ObjectListing objectListing = s3.listObjects(new ListObjectsRequest()
            .withBucketName(bucketName)
            .withPrefix("My"));

    s3.deleteObject(bucketName, key);
    s3.deleteBucket(bucketName);

} catch (AmazonServiceException ase) {
   // error handling...
} catch (AmazonClientException ace) {
   // error handling...
}
```

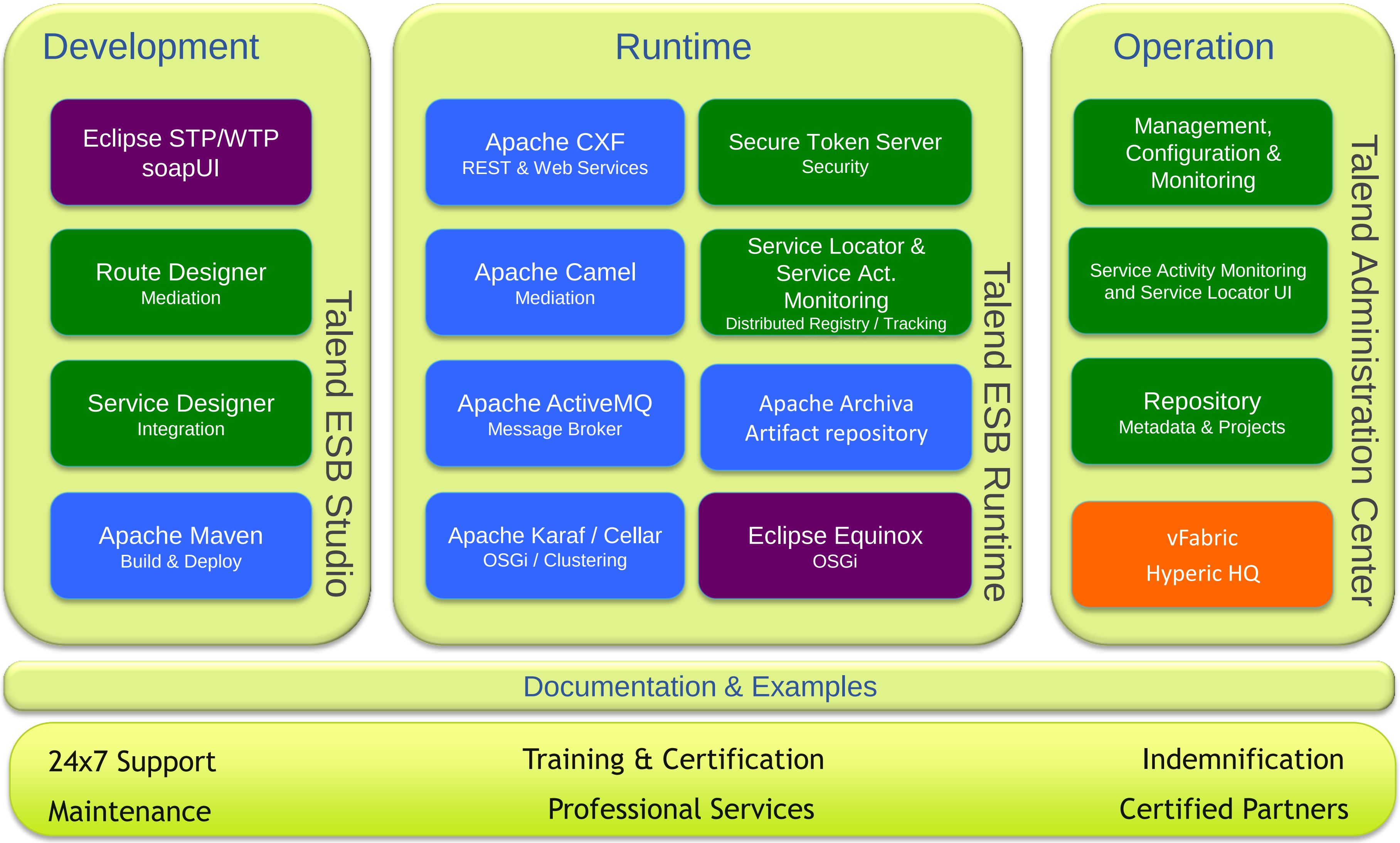
Code example: camel-aws component

```
// Producer
from(„jms:toS3Queue")
    .setHeader(S3Constants.KEY, simple(“order.txt"))
    .to("aws-s3://myBucket?accessKey=" + a + "&secretKey= " + s)


// Consumer
from("aws-s3://myBucket?accessKey=“ + a + "&secretKey=" + s)
   .to("log:S3logging")
```

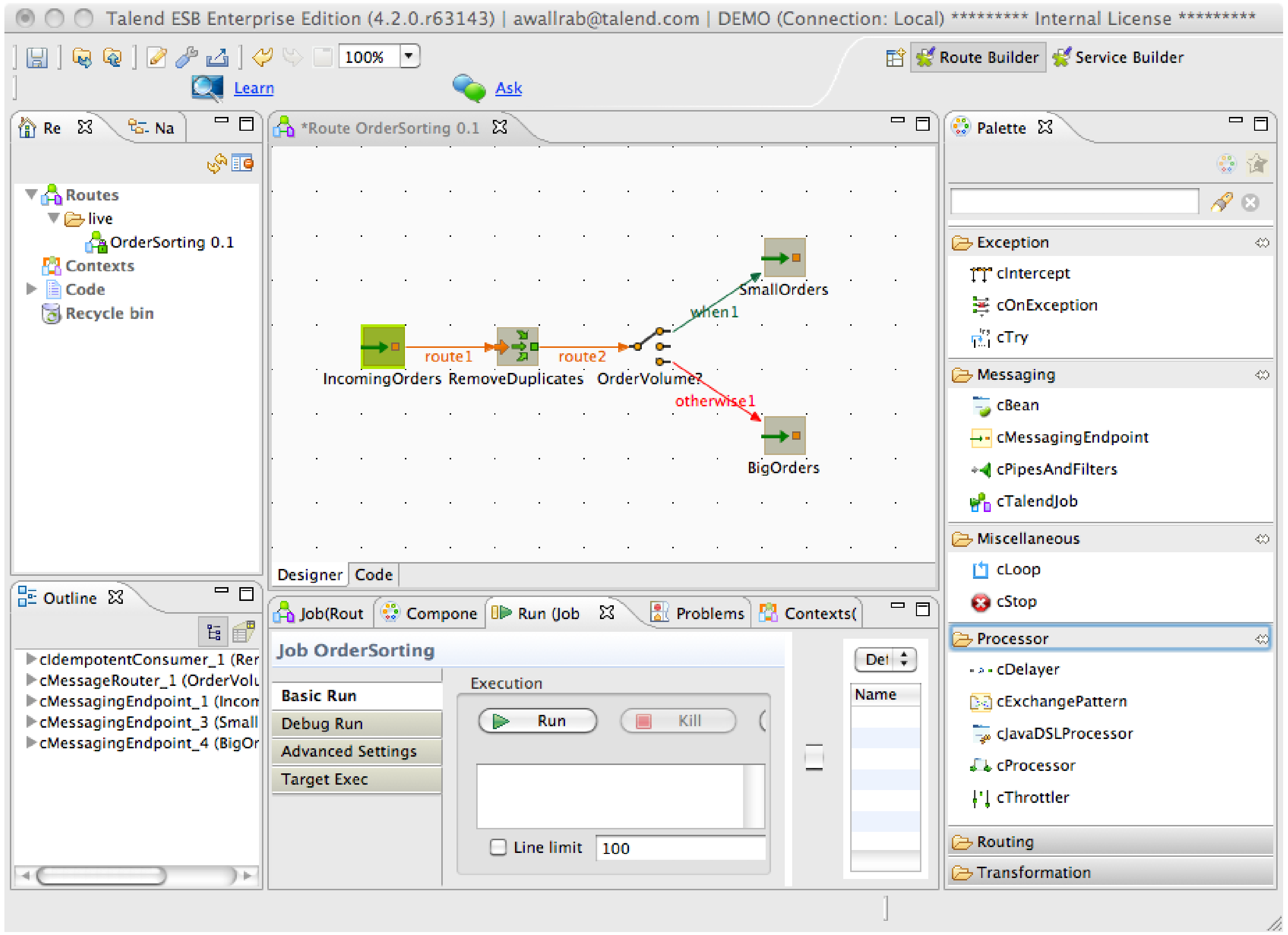
Tooling on top of Camel: Talend ESB



Tooling on top of Camel: Talend ESB



Tooling on top of Camel: Talend ESB



Route Builder

- Endpoints
- EIPs
- Processors
- Custom components

Configuration

- Components
- Endpoints

Code Generation

- 100% Java
- Camel Code
- Packaged as OSGi Bundles

Execution in the IDE

- Debugging
- Live statistics
- Short dev cycles

Live demo

amazon webservices™

S3 Simple Storage Service

Integration of a key-value database in action…

Alternative to Vendor APIs

➜ Generic APIs

jclouds

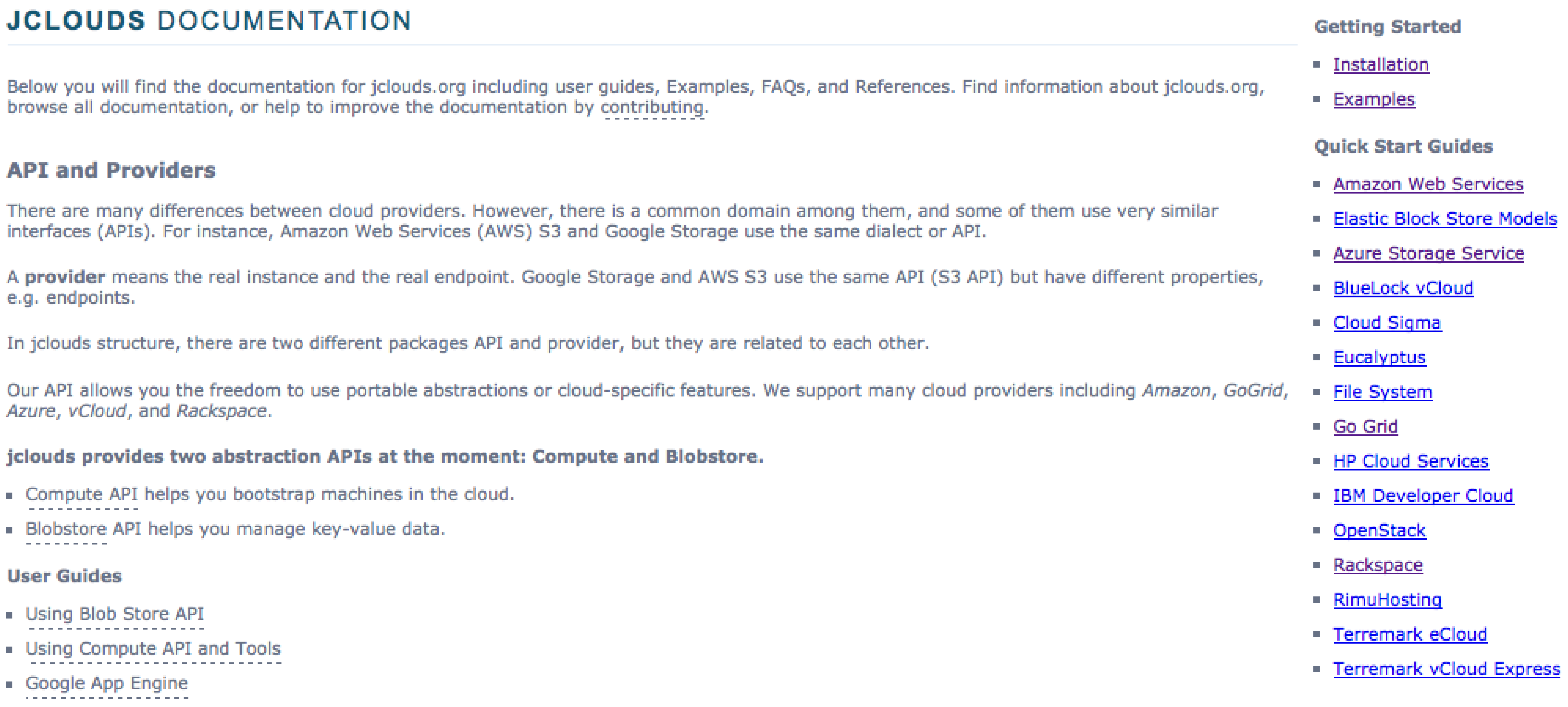
δ·CLOUD

sharpbox

a cloud storage programming interface

jClouds (Generic API)

Generic API for IaaS



JCLOUDS DOCUMENTATION

Below you will find the documentation for jclouds.org including user guides, Examples, FAQs, and References. Find information about jclouds.org, browse all documentation, or help to improve the documentation by contributing.

API and Providers

There are many differences between cloud providers. However, there is a common domain among them, and some of them use very similar interfaces (APIs). For instance, Amazon Web Services (AWS) S3 and Google Storage use the same dialect or API.

A **provider** means the real instance and the real endpoint. Google Storage and AWS S3 use the same API (S3 API) but have different properties, e.g. endpoints.

In jclouds structure, there are two different packages API and provider, but they are related to each other.

Our API allows you the freedom to use portable abstractions or cloud-specific features. We support many cloud providers including *Amazon*, *GoGrid*, *Azure*, *vCloud*, and *Rackspace*.

jclouds provides two abstraction APIs at the moment: Compute and Blobstore.

- Compute API helps you bootstrap machines in the cloud.
- Blobstore API helps you manage key-value data.

User Guides

- Using Blob Store API
- Using Compute API and Tools
- Google App Engine

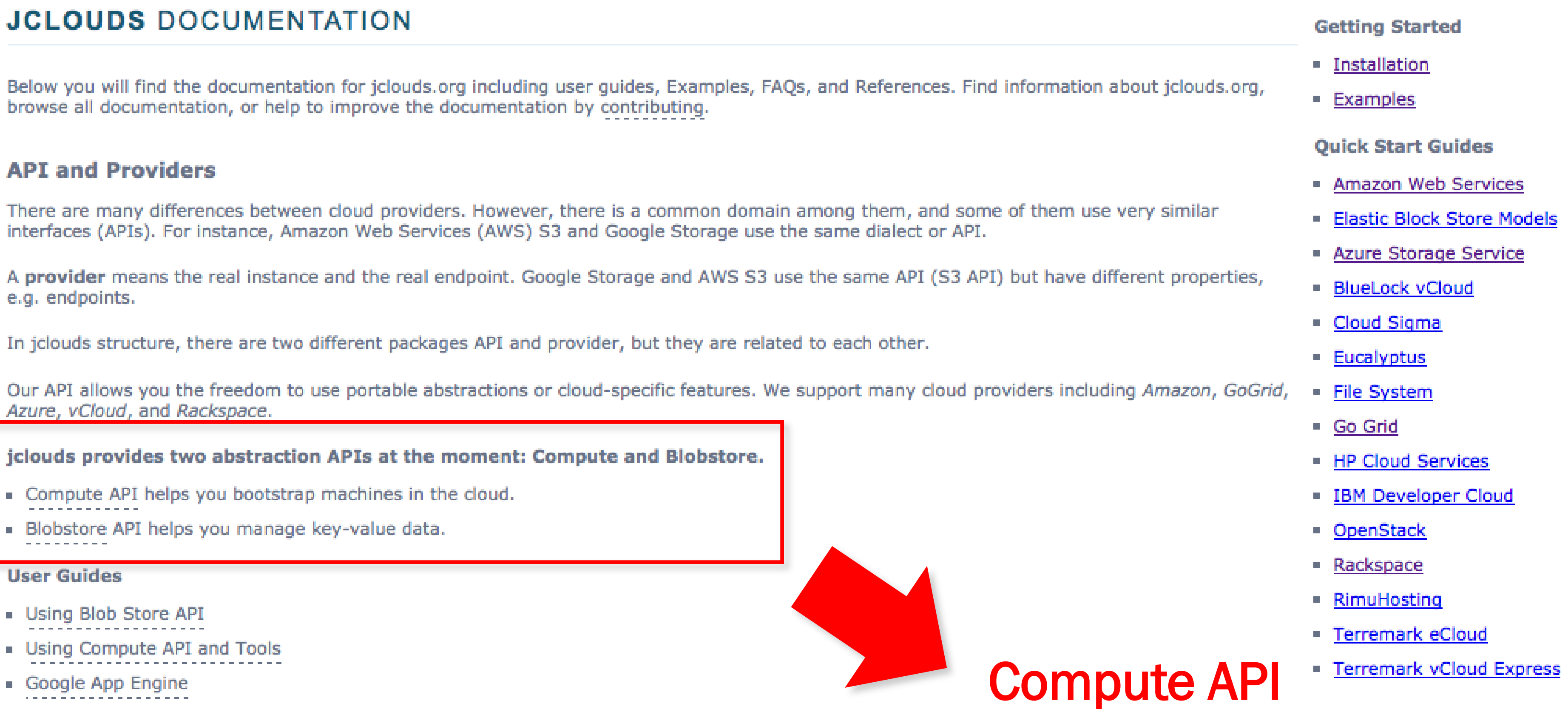
Getting Started

- Installation
- Examples

Quick Start Guides

- Amazon Web Services
- Elastic Block Store Models
- Azure Storage Service
- BlueLock vCloud
- Cloud Sigma
- Eucalyptus
- File System
- Go Grid
- HP Cloud Services
- IBM Developer Cloud
- OpenStack
- Rackspace
- RimuHosting
- Terremark eCloud
- Terremark vCloud Express

jClouds (Generic API)



JCLOUDS DOCUMENTATION

Below you will find the documentation for jclouds.org including user guides, Examples, FAQs, and References. Find information about jclouds.org, browse all documentation, or help to improve the documentation by contributing.

API and Providers

There are many differences between cloud providers. However, there is a common domain among them, and some of them use very similar interfaces (APIs). For instance, Amazon Web Services (AWS) S3 and Google Storage use the same dialect or API.

A **provider** means the real instance and the real endpoint. Google Storage and AWS S3 use the same API (S3 API) but have different properties, e.g. endpoints.

In jclouds structure, there are two different packages API and provider, but they are related to each other.

Our API allows you the freedom to use portable abstractions or cloud-specific features. We support many cloud providers including *Amazon*, *GoGrid*, *Azure*, *vCloud*, and *Rackspace*.

jclouds provides two abstraction APIs at the moment: Compute and Blobstore.

- Compute API helps you bootstrap machines in the cloud.
- Blobstore API helps you manage key-value data.

User Guides

- Using Blob Store API
- Using Compute API and Tools
- Google App Engine

Getting Started

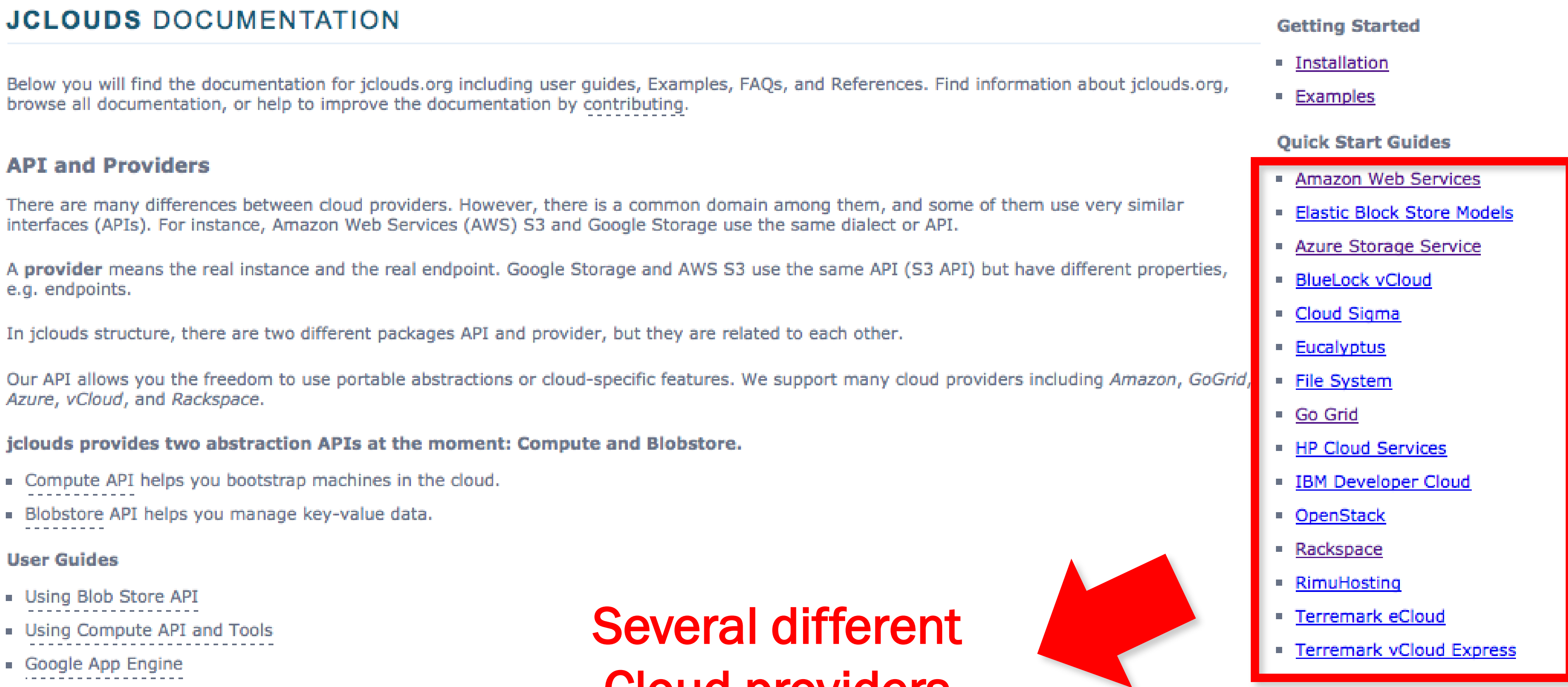
- Installation
- Examples

Quick Start Guides

- Amazon Web Services
- Elastic Block Store Models
- Azure Storage Service
- BlueLock vCloud
- Cloud Sigma
- Eucalyptus
- File System
- Go Grid
- HP Cloud Services
- IBM Developer Cloud
- OpenStack
- Rackspace
- RimuHosting
- Terremark eCloud
- Terremark vCloud Express

Compute API
Blobstore API

jClouds (Generic API)



JCLOUDS DOCUMENTATION

Below you will find the documentation for jclouds.org including user guides, Examples, FAQs, and References. Find information about jclouds.org, browse all documentation, or help to improve the documentation by contributing.

API and Providers

There are many differences between cloud providers. However, there is a common domain among them, and some of them use very similar interfaces (APIs). For instance, Amazon Web Services (AWS) S3 and Google Storage use the same dialect or API.

A **provider** means the real instance and the real endpoint. Google Storage and AWS S3 use the same API (S3 API) but have different properties, e.g. endpoints.

In jclouds structure, there are two different packages API and provider, but they are related to each other.

Our API allows you the freedom to use portable abstractions or cloud-specific features. We support many cloud providers including *Amazon*, *GoGrid*, *Azure*, *vCloud*, and *Rackspace*.

jclouds provides two abstraction APIs at the moment: Compute and Blobstore.

- Compute API helps you bootstrap machines in the cloud.
- Blobstore API helps you manage key-value data.

User Guides

- Using Blob Store API
- Using Compute API and Tools
- Google App Engine

Getting Started

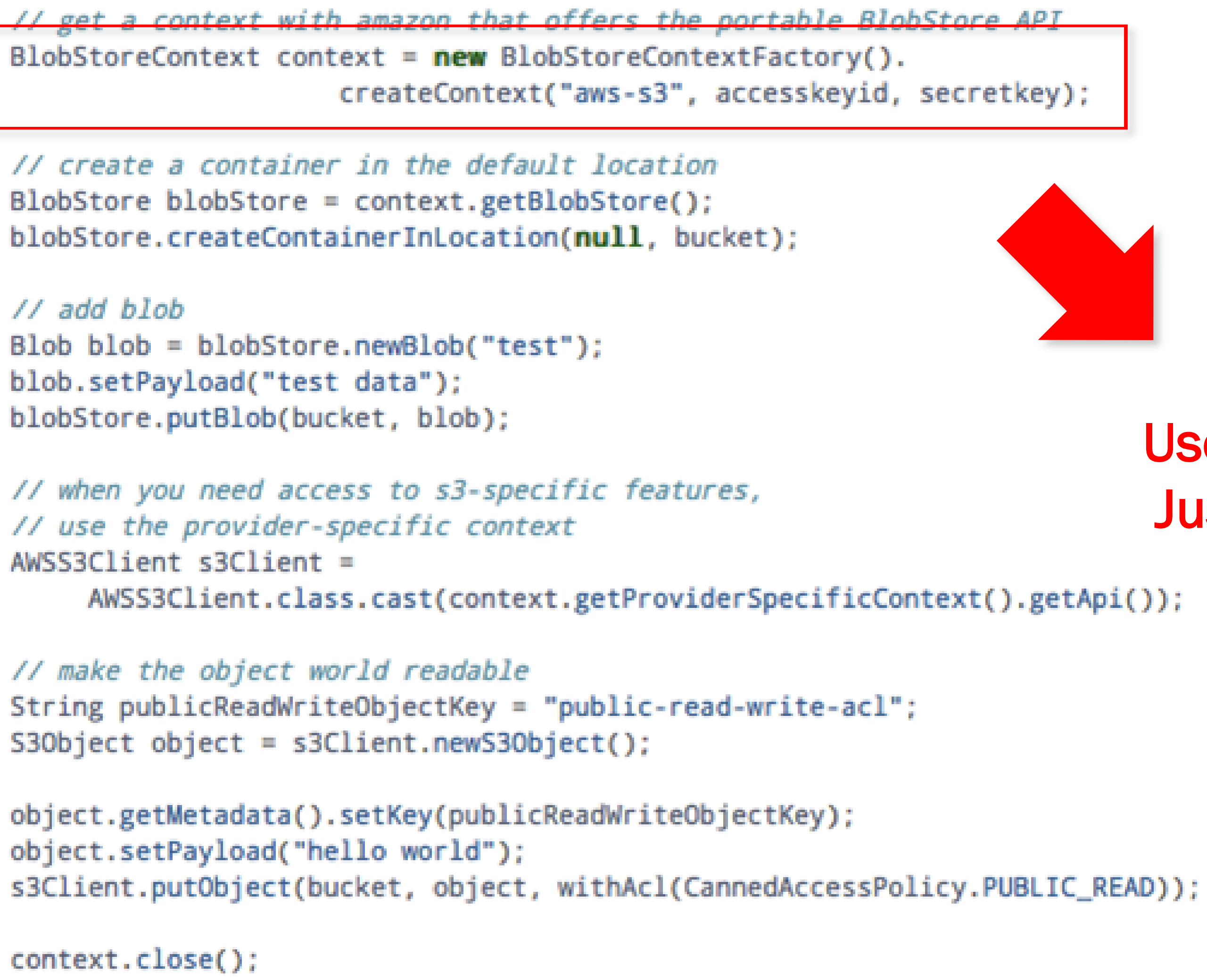
- Installation
- Examples

Quick Start Guides

- Amazon Web Services
- Elastic Block Store Models
- Azure Storage Service
- BlueLock vCloud
- Cloud Sigma
- Eucalyptus
- File System
- Go Grid
- HP Cloud Services
- IBM Developer Cloud
- OpenStack
- Rackspace
- RimuHosting
- Terremark eCloud
- Terremark vCloud Express

Several different Cloud providers supported

jClouds (Generic API) – AWS S3 Blobstore (Java)



```java
// get a context with amazon that offers the portable BlobStore API
BlobStoreContext context = new BlobStoreContextFactory().
                    createContext("aws-s3", accesskeyid, secretkey);

// create a container in the default location
BlobStore blobStore = context.getBlobStore();
blobStore.createContainerInLocation(null, bucket);

// add blob
Blob blob = blobStore.newBlob("test");
blob.setPayload("test data");
blobStore.putBlob(bucket, blob);

// when you need access to s3-specific features,
// use the provider-specific context
AWSS3Client s3Client =
     AWSS3Client.class.cast(context.getProviderSpecificContext().getApi());

// make the object world readable
String publicReadWriteObjectKey = "public-read-write-acl";
S3Object object = s3Client.newS3Object();

object.getMetadata().setKey(publicReadWriteObjectKey);
object.setPayload("hello world");
s3Client.putObject(bucket, object, withAcl(CannedAccessPolicy.PUBLIC_READ));

context.close();
```

Use another provider?
Just change this line!

Code example: camel-jclouds component

```
 from("direct:toJcloudsAwsS3")
    .setHeader(JcloudsConstants.BLOB_NAME,  "jclouds-demo-tutorial.txt")
    .setHeader(JcloudsConstants.CONTAINER_NAME,  "kw-s3-data")
.to("jclouds:blobstore:aws-s3")



from("direct:toJcloudsMicrosoftAzure")
    .setHeader(JcloudsConstants.BLOB_NAME,  "jclouds-demo-tutorial.txt")
    .setHeader(JcloudsConstants.CONTAINER_NAME, "kw-s3-data")
.to("jclouds:blobstore:azureblob")
```

Agenda

1) Introduction to NoSQL
2) Introduction to Apache Camel
3) Integration of a Document-oriented Database
4) Integration of a Key-Value Database
5) Integration of an In-Memory Database
6) Integration of a Graph-oriented Database
7) Integration of a Column-oriented Database
8) Custom NoSQL Components

In-memory database

HAZELCAST SOFTWARE

ORACLE Coherence

SAP HANA

GIGASPACES

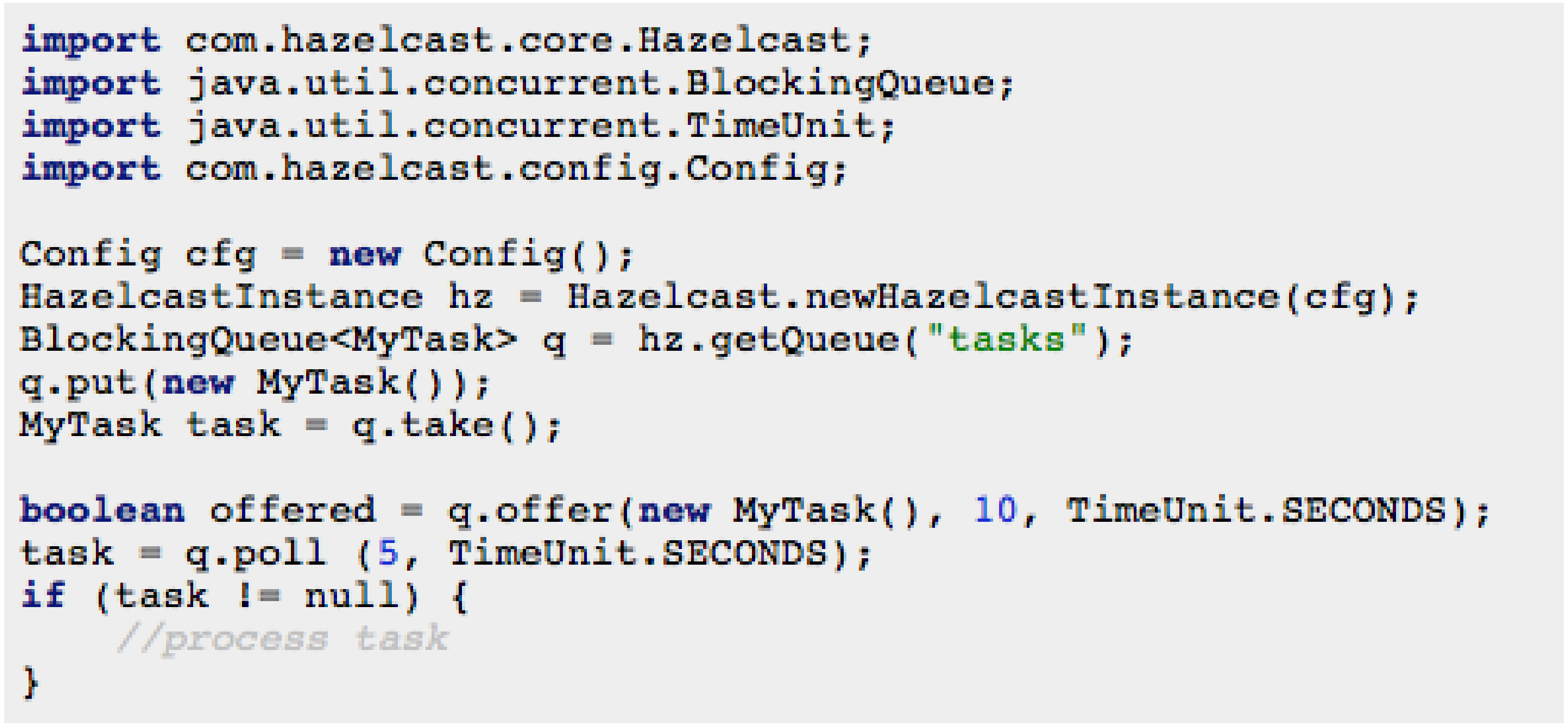
EHCACHE a product from TERRACOTTA

In-memory database

HAZELCAST SOFTWARE

- In-memory data grid
- Clustering and highly scalable data distribution solution for Java platform
- Architecture is peer-to-peer
- Distributed Java data structures (Queue, Set, List, Map, Lock, Topic)
- Java and REST API

Code example: Hazelcast Java API



```java
import com.hazelcast.core.Hazelcast;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.TimeUnit;
import com.hazelcast.config.Config;

Config cfg = new Config();
HazelcastInstance hz = Hazelcast.newHazelcastInstance(cfg);
BlockingQueue<MyTask> q = hz.getQueue("tasks");
q.put(new MyTask());
MyTask task = q.take();

boolean offered = q.offer(new MyTask(), 10, TimeUnit.SECONDS);
task = q.poll (5, TimeUnit.SECONDS);
if (task != null) {
    //process task
}
```

Code example: camel-hazelcast component

```
// Producer
from("direct:add")
      .setHeader(HazelcastConstants.OPERATION, „add“)
      .to("hazelcast:queue:foo");

// Consumer
from("hazelcast:queue:foo")
      .log(“content of object foo: ${body}");
```

Live demo

HAZELCAST SOFTWARE

Integration of an in-memory database in action…

Agenda

1) Introduction to NoSQL
2) Introduction to Apache Camel
3) Integration of a Document-oriented Database
4) Integration of a Key-Value Database
5) Integration of an In-Memory Database
6) Integration of a Graph-oriented Database
7) Integration of a Column-oriented Database
8) Custom NoSQL Components

Graph-oriented database

Neo4j
the graph database

*dex

sones

twitter / flockdb

InfiniteGraph
The Distributed Graph Database

Graph-oriented database

- Neo Technology
- Graphs rather than tables
- Nodes, edges, and properties to represent and store data
- Index-free adjacency
- REST API and many SDKs (Java, .NET, Ruby, PHP, Python, etc.)
- Embedded, disk-based, fully transactional
- Powerful tool for graph-like queries
- Example: Facebook friends

Code example: Neo4j Ruby API

```ruby
require 'rubygems'
require 'neography'

@neo = Neography::Rest.new

def create_person(name)
  @neo.create_node("name" => name)
end

def make_mutual_friends(node1, node2)
  @neo.create_relationship("friends", node1, node2)
  @neo.create_relationship("friends", node2, node1)
end

def suggestions_for(node)
  @neo.traverse(node,"nodes", {"order" => "breadth first",
                                          "uniqueness" => "node global",
                                          "relationships" => {"type"=> "friends", "direction" => "in"},
                                          "return filter" => {
                                            "language" => "javascript",
                                            "body" => "position.length() == 2;"},
                                          "depth" => 2})
end

johnathan = create_person('Johnathan')
mark      = create_person('Mark')
phill     = create_person('Phill')
mary      = create_person('Mary')
luke      = create_person('Luke')

make_mutual_friends(johnathan, mark)
make_mutual_friends(mark, mary)
make_mutual_friends(mark, phill)
make_mutual_friends(phill, mary)
make_mutual_friends(phill, luke)

puts "Johnathan should become friends with #{suggestions_for(johnathan).map{|n| n["data"]["name"]}.join(', ')}"

# RESULT
# Johnathan should become friends with Mary, Phill
```

Code example: camel-neo4j component

```
// Producer
 from("jms:createNewNeo4jNode")
          .to("neo4j:http://Neo4jServer:7474/data");


// Consumer
from(„neo4j://todo)...
```

Not implemented in current Camel release (2.11) ☹

→ Use Camel‘s REST components (shown in some minutes...)

Live demo

Neo4j
the graph database

Integration of a graph-oriented database in action…

Agenda

1) Introduction to NoSQL
2) Introduction to Apache Camel
3) Integration of a Document-oriented Database
4) Integration of a Key-Value Database
5) Integration of an In-Memory Database
6) Integration of a Graph-oriented Database
7) Integration of a Column-oriented Database
8) Custom NoSQL Components

Column-oriented database

APACHE HBASE

MONETDB

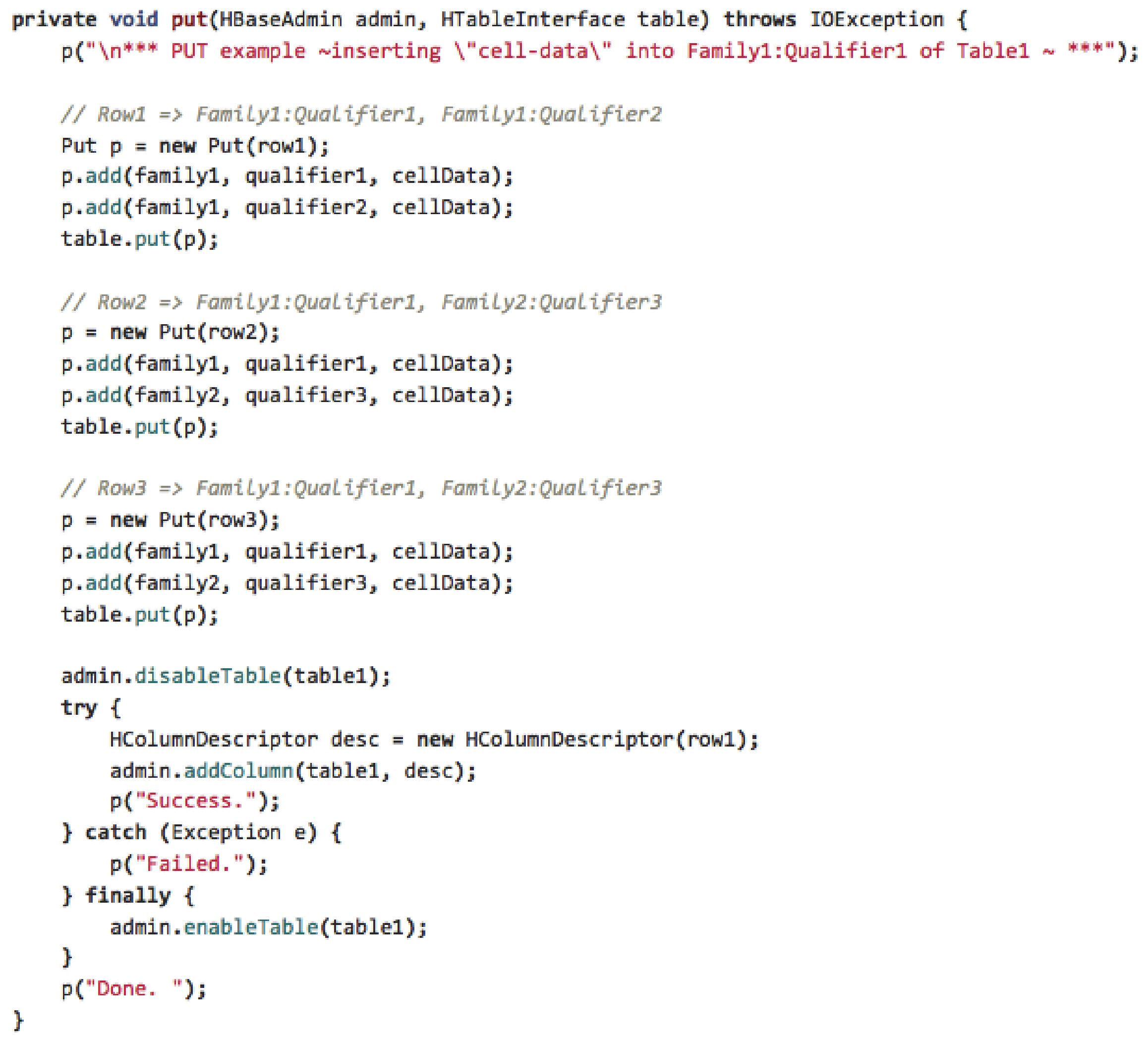
Google Bigtable

Cassandra

HBase

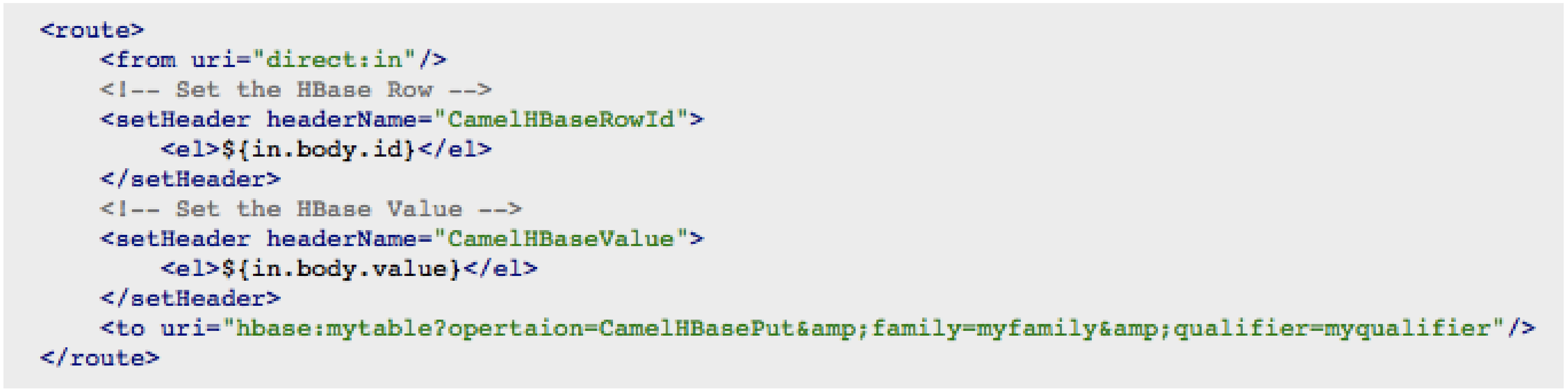
- Modeled after Google's BigTable
- Runs on top of HDFS (Hadoop Distributed Filesystem)
- Can serve as the input and output for MapReduce jobs run in Hadoop
- Stores data tables as sections of columns of data rather than as rows of data
- Java API plus REST, Avro or Thrift gateway APIs
- Use HBase when you need random, realtime read/write access to your Big Data
- Example: Advantages for DWHs, CRMs, and other ad-hoc inquiry systems where aggregates are computed over large numbers of similar data items.

Code example: HBase Java API



```
private void put(HBaseAdmin admin, HTableInterface table) throws IOException {
    p("\n*** PUT example ~inserting \"cell-data\" into Family1:Qualifier1 of Table1 ~ ***");

    // Row1 => Family1:Qualifier1, Family1:Qualifier2
    Put p = new Put(row1);
    p.add(family1, qualifier1, cellData);
    p.add(family1, qualifier2, cellData);
    table.put(p);

    // Row2 => Family1:Qualifier1, Family2:Qualifier3
    p = new Put(row2);
    p.add(family1, qualifier1, cellData);
    p.add(family2, qualifier3, cellData);
    table.put(p);

    // Row3 => Family1:Qualifier1, Family2:Qualifier3
    p = new Put(row3);
    p.add(family1, qualifier1, cellData);
    p.add(family2, qualifier3, cellData);
    table.put(p);

    admin.disableTable(table1);
    try {
        HColumnDescriptor desc = new HColumnDescriptor(row1);
        admin.addColumn(table1, desc);
        p("Success.");
    } catch (Exception e) {
        p("Failed.");
    } finally {
        admin.enableTable(table1);
    }
    p("Done. ");
}
```

Code example: camel-hbase component



```
<route>
    <from uri="direct:in"/>
    <!-- Set the HBase Row -->
    <setHeader headerName="CamelHBaseRowId">
        <el>${in.body.id}</el>
    </setHeader>
    <!-- Set the HBase Value -->
    <setHeader headerName="CamelHBaseValue">
        <el>${in.body.value}</el>
    </setHeader>
    <to uri="hbase:mytable?opertaion=CamelHBasePut&amp;family=myfamily&amp;qualifier=myqualifier"/>
</route>
```

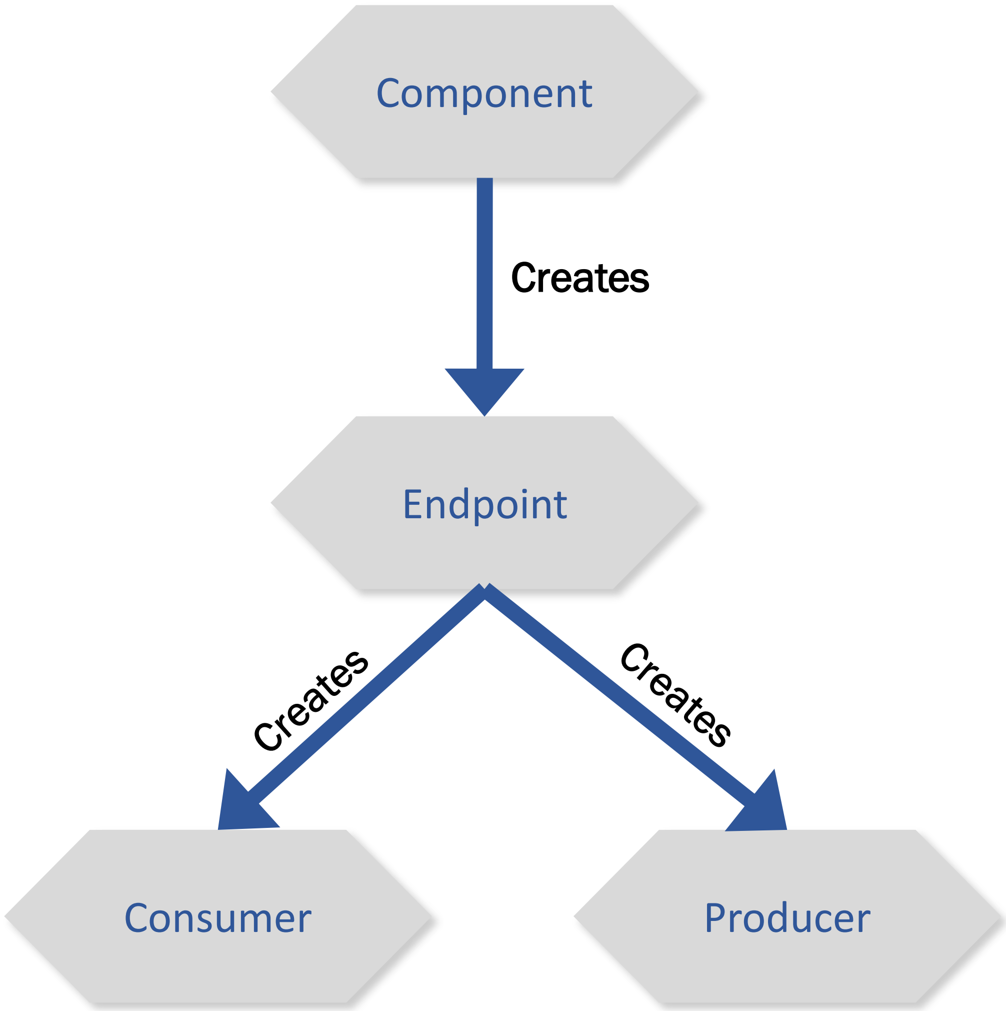
Live demo

Integration of a column-oriented database in action…

Agenda

1) Introduction to NoSQL
2) Introduction to Apache Camel
3) Integration of a Document-oriented Database
4) Integration of a Key-Value Database
5) Integration of an In-Memory Database
6) Integration of a Graph-oriented Database
7) Integration of a Column-oriented Database
8) Custom NoSQL Components

Custom NoSQL components



Live demo

Custom NoSQL components in action…

Alternative for custom NoSQL components

Sluggish Boy ???

- SOAP
- REST

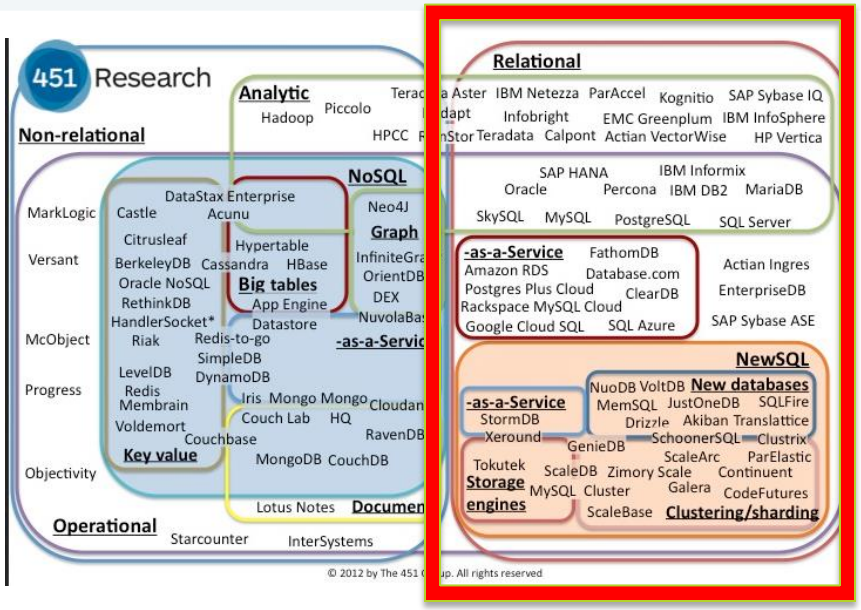
Code example: REST API for Salesforce object store

```
// Salesforce Query (SOQL) via REST API
from("direct:salesforceViaHttpLIST")
        .setHeader("X-PrettyPrint", 1)
        .setHeader("Authorization", accessToken)
        .setHeader(Exchange.CONTENT_TYPE, "application/json")
.to("https://na14.salesforce.com/services/data/v20.0/query?q=SELECT+name+from
        +Article__c")


// Salesforce CREATE via REST API
 from("direct:salesforceViaHttpCREATE")
        .setHeader("X-PrettyPrint", 1)
        .setHeader("Authorization", accessToken)
        .setHeader(Exchange.CONTENT_TYPE, "application/json“)
.to("https://na14.salesforce.com/services/data/v20.0/sobjects/Article__c")
```

Live demo

NoSQL integration via REST in action…

SQL is still very alive, of course…



Java Database Programming With JDBC

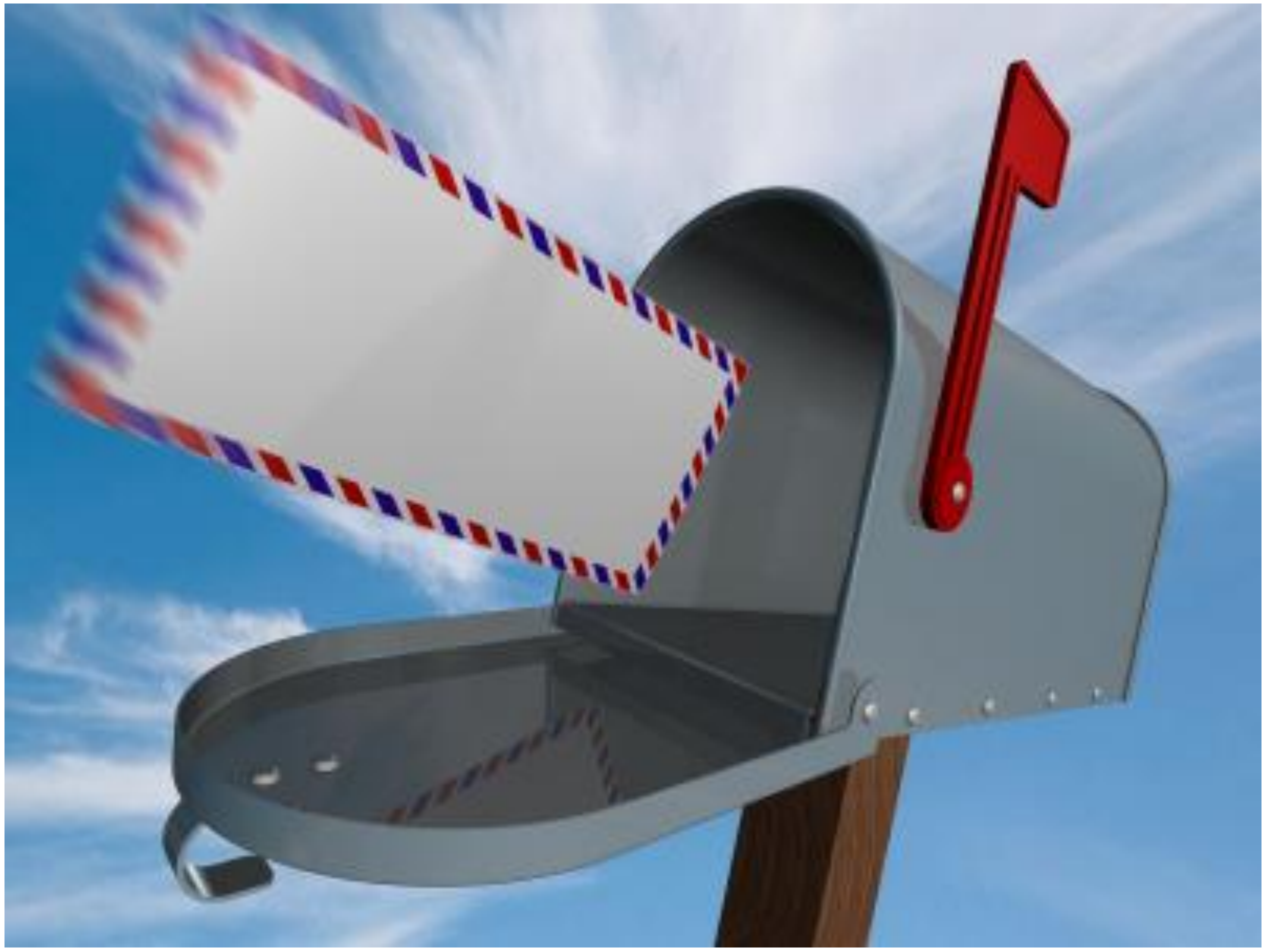
mybatis SQL Mapping Framework for Java

HIBERNATE

Camel SQL components:

- sql:statement
- jdbc:dataSourceName
- jpa://entityName
- mybatis://statementName
- hibernate://entityName

Did you get the key message?



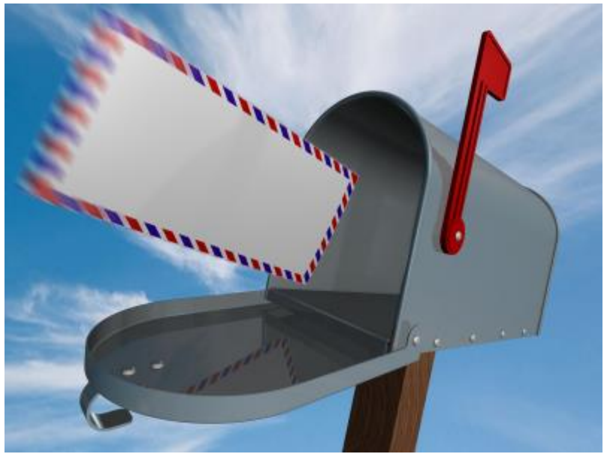
Key messages

NoSQL cannot be avoided, and must be integrated! ✓

NoSQL integration is already possible! ✓

Apache Camel helps a lot! ✓

Did you get the key message?



Thank you for your attention. Questions?

kwaehner@talend.com
www.kai-waehner.de
LinkedIn / Xing
@KaiWaehner