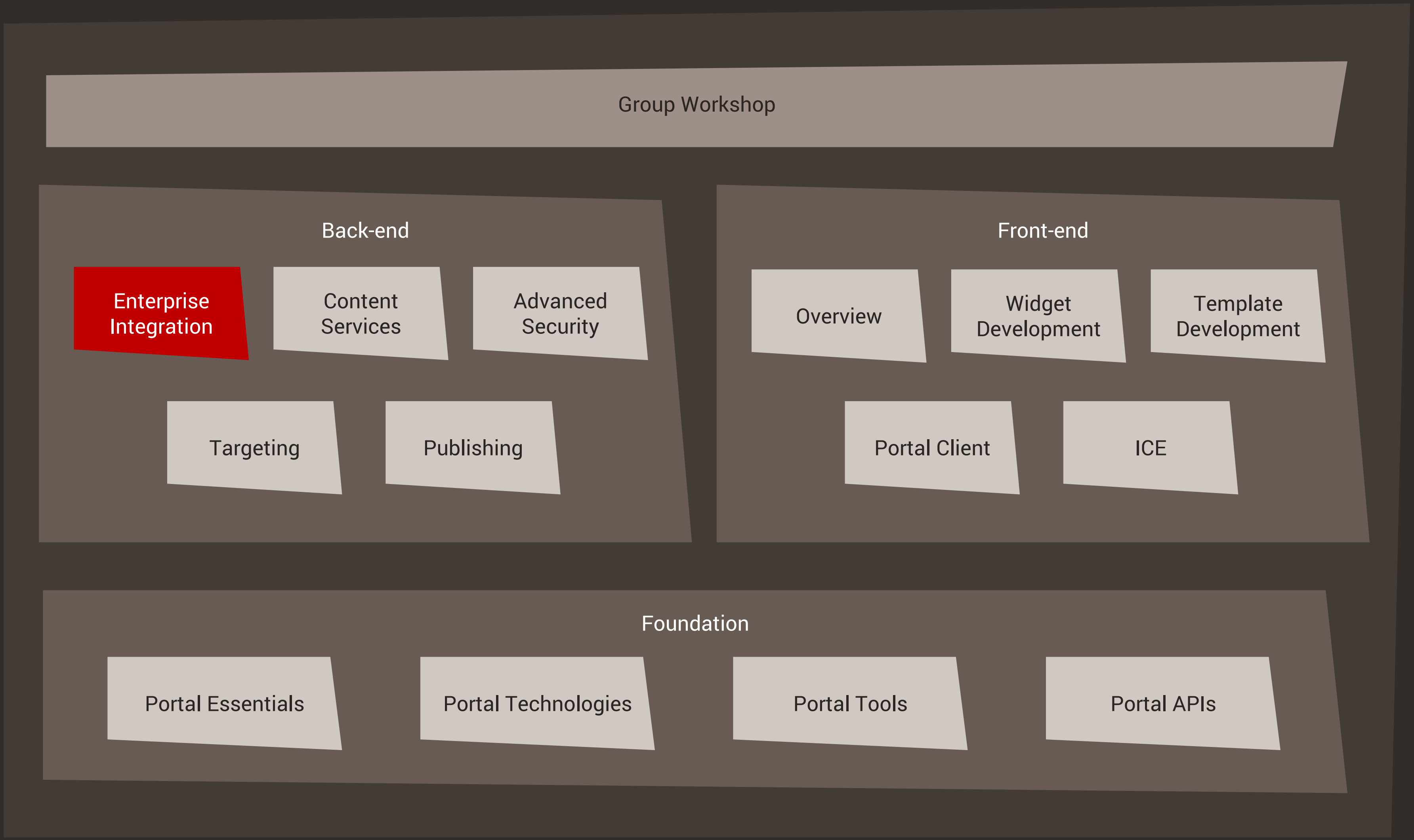
BACKBASE

# Enterprise Integration

## Focus: Backend

# Training Overview



Group Workshop

Back-end

- Enterprise Integration
- Content Services
- Advanced Security
- Targeting
- Publishing

Front-end

- Overview
- Widget Development
- Template Development
- Portal Client
- ICE

Foundation

- Portal Essentials
- Portal Technologies
- Portal Tools
- Portal APIs

# Objectives

1. Enterprise integration
2. Introduction to Apache Camel
3. Apache Camel in CXP
4. Exercises

# Enterprise Integration

What is it and why do we need it

# Challenges

- Data integration
  - Web Services
  - REST
  - Messaging
  - Proprietary API
- HTML integration
  - Proxy
  - Screen scraping
- Caching data

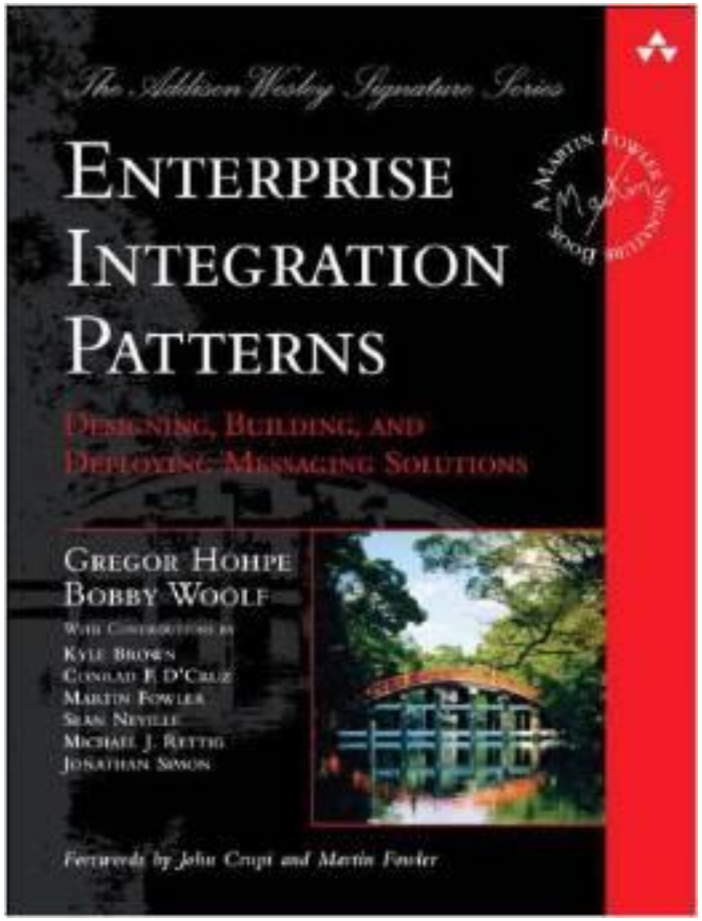
# Challenges

- Filter and transform data
  - Convert POJO’s to JSON
  - Combine data sources
  - Renditions
    - PDF
    - Excel
    - HTML
- Route messages
  - Send messages to different targets
    - Based on payloads
    - Or other rules

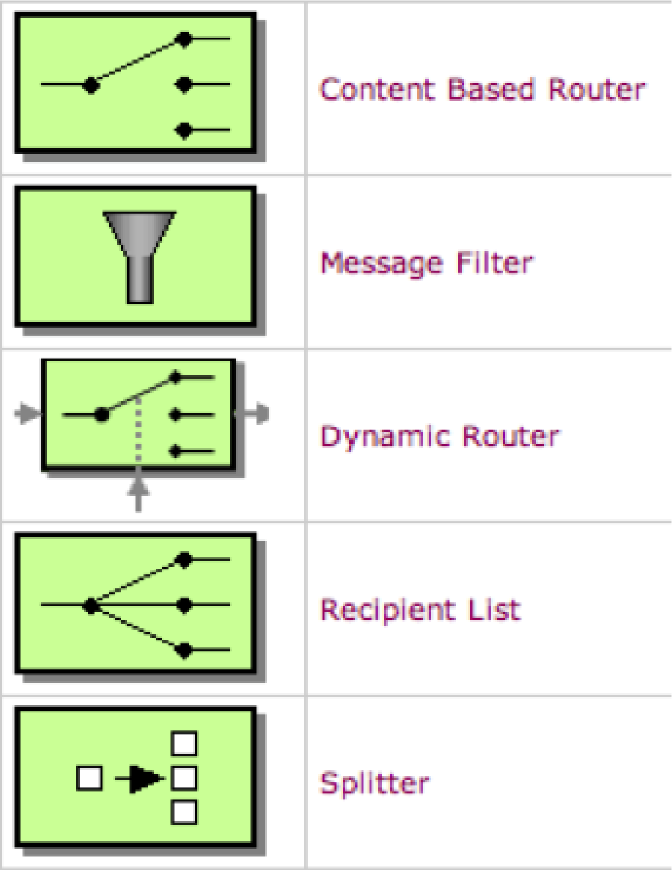
# Enterprise Integration Patterns

- Patterns
  - Provide guidance
  - Common language
  - Outlines solutions to common challenges
  - 65+ patterns

- Available platforms
  - Spring Integration
  - Mule ESB
  - Apache Camel
  - Commercial offerings


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

# Enterprise Integration Patterns



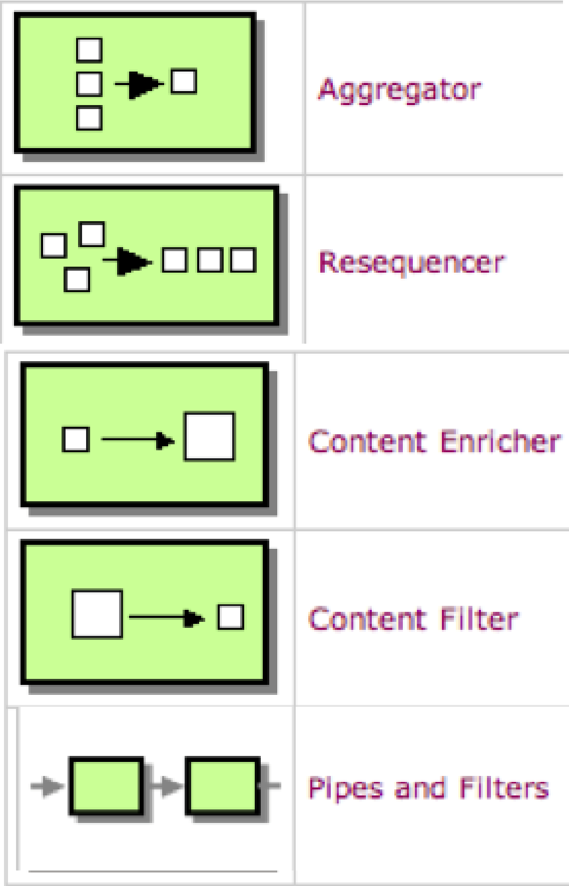
Content Based Router

Message Filter

Dynamic Router

Recipient List

Splitter



Aggregator

Resequencer

Content Enricher

Content Filter

Pipes and Filters

http://camel.apache.org/eip

# Introduction to Apache Camel

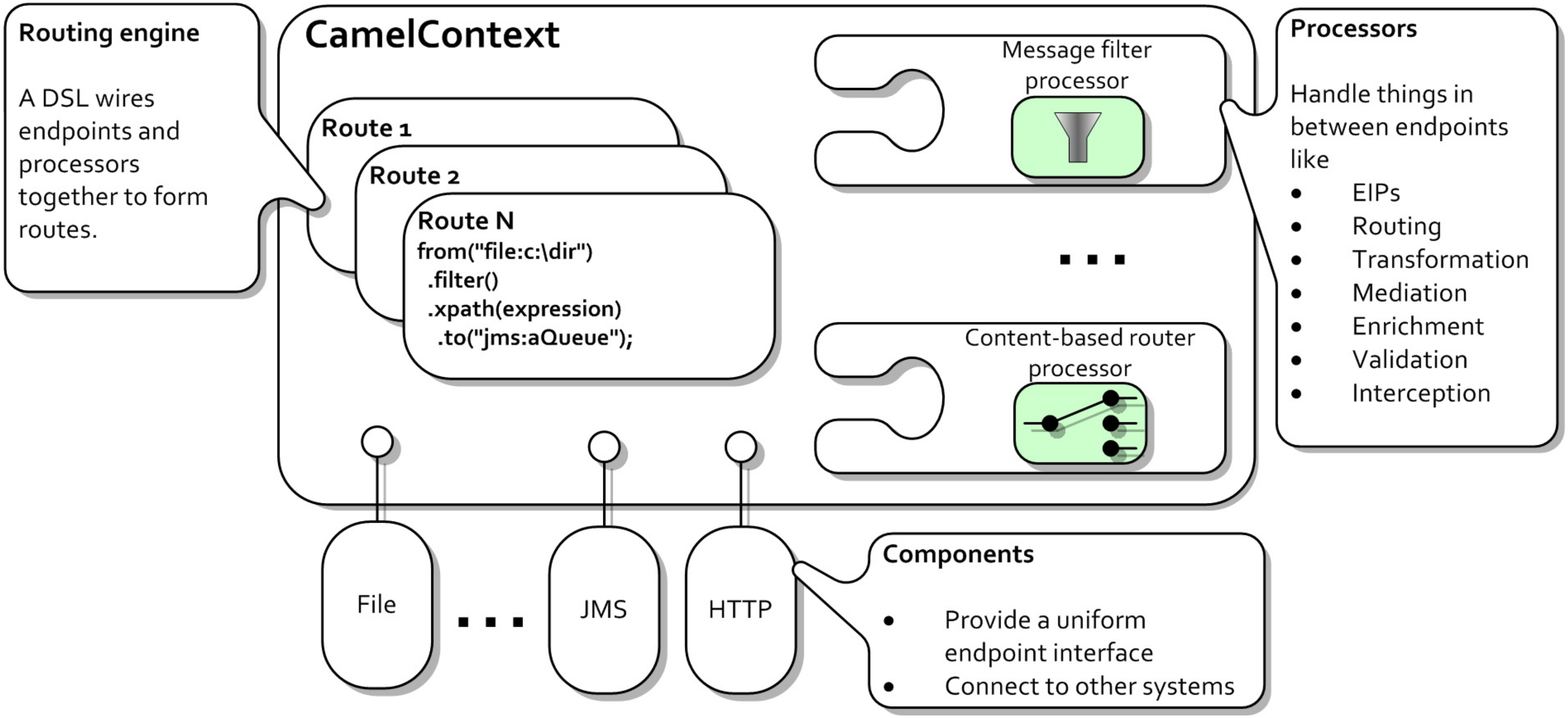
Heavy payloads with few resources

Apache Camel™

- Light weight integration library
- Kind of an embedded ESB
- Enterprise Integration Patterns
- Domain Specific Language
- Routing and Mediation
- Components
- Built in Transformers
- Active Community
- Easy to extend

**Concise**
**Application**
**Messaging**
**Exchange**
**Language**

# Quick Overview



```
from("file:c:\dir")
 .filter()
 .xpath(expression)
  .to("jms:aQueue");
```

**Routing engine**

A DSL wires endpoints and processors together to form routes.

**Processors**

Handle things in between endpoints like

- EIPs
- Routing
- Transformation
- Mediation
- Enrichment
- Validation
- Interception

**Components**

- Provide a uniform endpoint interface
- Connect to other systems

# Endpoint

- Describes a service from which information can be retrieved or sent to
- Example endpoints
  - JMS queue
  - Web Service
  - FTP Server
  - Twitter
  - Plain Old Java Object
- Endpoints can behave in 2 ways
  - Producer
  - Consumer

# Component

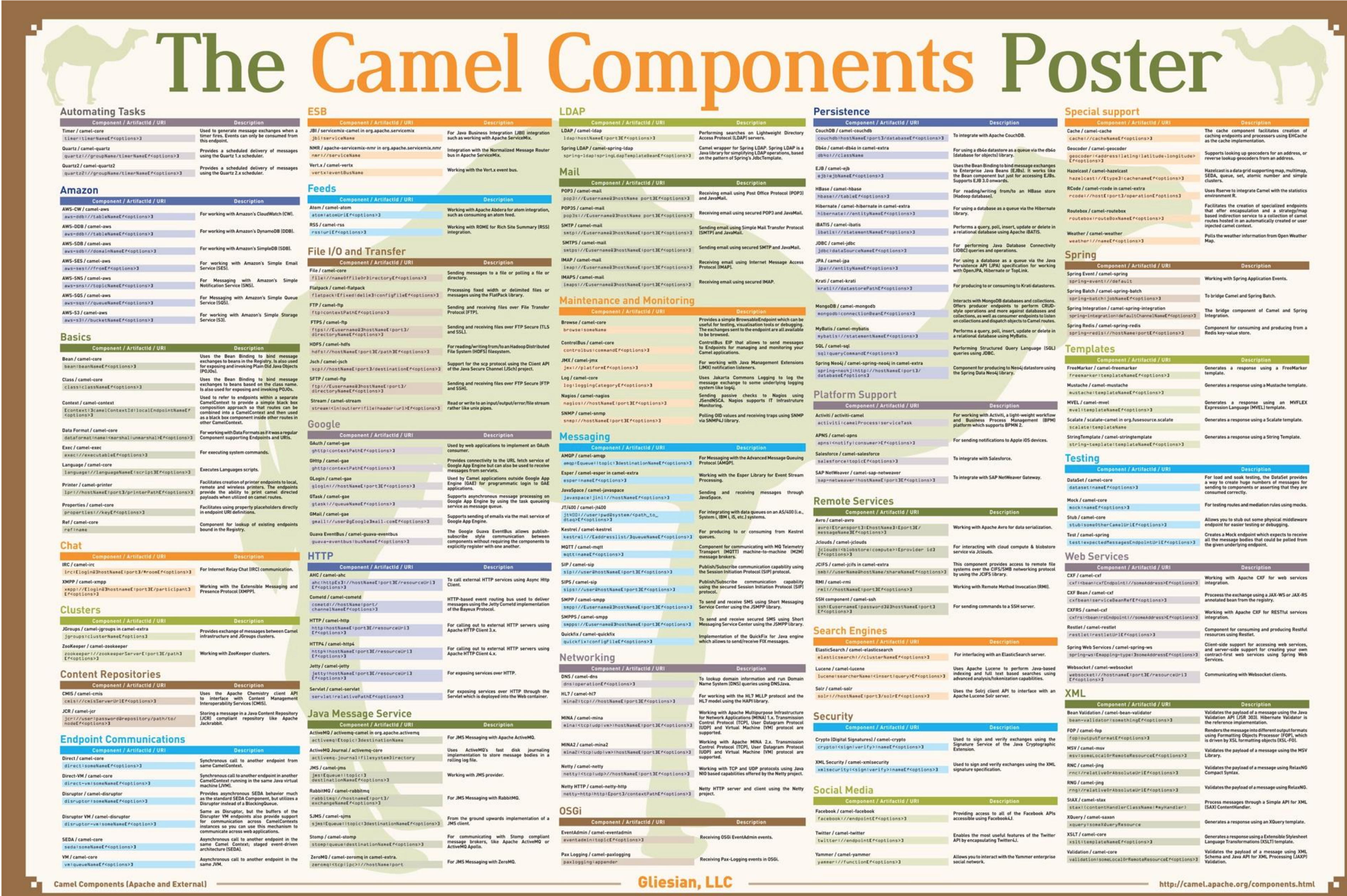
- AKA: EndpointFactory
- Component creates Endpoints based on configuration
- Identified by using an URI format
  - `scheme:localpart[?options]`
- Creates consumer and producer endpoints

```
<from uri="servlet:///sushi/html?matchOnUriPrefix=true&amp;httpBindingRef=#sushiHttpBinding&amp;servletName=Integration Servlet");

<to uri="https4://www.backbase.com?throwExceptionOnFailure=false&amp;httpClient.handleRedirects=true&amp;httpClient.maxRedirects=3"/>
```

- Lots of Camel Components available
  - Full list at: http://camel.apache.org/components.html

BACKBASE

# Apache Camel



# The Camel Components Poster

## Automating Tasks

| Component / ArtifactId / URI | Description |
|---|---|
| Timer / camel-core<br>`timer:timerName[?<options>]` | Used to generate message exchanges when a timer fires. Events can only be consumed from this endpoint. |
| Quartz / camel-quartz<br>`quartz://groupName/timerName[?<options>]` | Provides a scheduled delivery of messages using the Quartz 1.x scheduler. |
| Quartz2 / camel-quartz2<br>`quartz2://groupName/timerName[?<options>]` | Provides a scheduled delivery of messages using the Quartz 2.x scheduler. |

## Amazon

| Component / ArtifactId / URI | Description |
|---|---|
| AWS-CW / camel-aws<br>`aws-ddb://tableName[?<options>]` | For working with Amazon's CloudWatch (CW). |
| AWS-DDB / camel-aws<br>`aws-ddb://tableName[?<options>]` | For working with Amazon's DynamoDB (DDB). |
| AWS-SDB / camel-aws<br>`aws-sdb://domainName[?<options>]` | For working with Amazon's SimpleDB (SDB). |
| AWS-SES / camel-aws<br>`aws-ses://from[?<options>]` | For working with Amazon's Simple Email Service (SES). |
| AWS-SNS / camel-aws<br>`aws-sns://topicName[?<options>]` | For Messaging with Amazon's Simple Notification Service (SNS). |
| AWS-SQS / camel-aws<br>`aws-sqs://queueName[?<options>]` | For Messaging with Amazon's Simple Queue Service (SQS). |
| AWS-S3 / camel-aws<br>`aws-s3://bucketName[?<options>]` | For working with Amazon's Simple Storage Service (S3). |

## Basics

| Component / ArtifactId / URI | Description |
|---|---|
| Bean / camel-core<br>`bean:beanName[?<options>]` | Uses the Bean Binding to bind message exchanges to beans in the Registry. Is also used for exposing and invoking Plain Old Java Objects (POJOs). |
| Class / camel-core<br>`class:className[?<options>]` | Uses the Bean Binding to bind message exchanges to beans based on the class name. Is also used for exposing and invoking POJOs. |
| Context / camel-context<br>`[context:]camelContextId:localEndpointName[?<options>]` | Used to refer to endpoints within a separate CamelContext to provide a simple black box composition approach so that routes can be combined into a CamelContext and then used as a black box component inside other routes in other CamelContext. |
| Data Format / camel-core<br>`dataformat:name:<marshal\|unmarshal>[?<options>]` | For working with Data Formats as if it was a regular Component supporting Endpoints and URIs. |
| Exec / camel-exec<br>`exec://executable[?<options>]` | For executing system commands. |
| Language / camel-core<br>`language://languageName[:script][?<options>]` | Executes Languages scripts. |
| Printer / camel-printer<br>`lpr://hostName[:port]/printerPath[?<options>]` | Facilitates creation of printer endpoints to local, remote and wireless printers. The endpoints provide the ability to print camel directed payloads when utilized on camel routes. |
| Properties / camel-core<br>`properties://key[?<options>]` | Facilitates using property placeholders directly in endpoint URI definitions. |
| Ref / camel-core<br>`ref:name` | Component for lookup of existing endpoints bound in the Registry. |

## Chat

| Component / ArtifactId / URI | Description |
|---|---|
| IRC / camel-irc<br>`irc:[login@]hostName[:port]/#room[?<options>]` | For Internet Relay Chat (IRC) communication. |
| XMPP / camel-xmpp<br>`xmpp://[login@]hostname[:port][/participant][?<options>]` | Working with the Extensible Messaging and Presence Protocol (XMPP). |

## Clusters

| Component / ArtifactId / URI | Description |
|---|---|
| JGroups / camel-jgroups in camel-extra<br>`jgroups:clusterName[?options]` | Provides exchange of messages between Camel infrastructure and JGroups clusters. |
| ZooKeeper / camel-zookeeper<br>`zookeeper://zookeeperServer[:port][/path][?<options>]` | Working with ZooKeeper clusters. |

## Content Repositories

| Component / ArtifactId / URI | Description |
|---|---|
| CMIS / camel-cmis<br>`cmis://cmisServerUrl[?<options>]` | Uses the Apache Chemistry client API to interface with Content Management Interoperability Services (CMIS). |
| JCR / camel-jcr<br>`jcr://user:password@repository/path/to/node[?<options>]` | Storing a message in a Java Content Repository (JCR) compliant repository like Apache Jackrabbit. |

## Endpoint Communications

| Component / ArtifactId / URI | Description |
|---|---|
| Direct / camel-core<br>`direct:someName[?<options>]` | Synchronous call to another endpoint from same CamelContext. |
| Direct-VM / camel-core<br>`direct-vm:someName[?<options>]` | Synchronous call to another endpoint in another CamelContext running in the same Java virtual machine (JVM). |
| Disruptor / camel-disruptor<br>`disruptor:someName[?<option>]` | Provides asynchronous SEDA behavior much as the standard SEDA Component, but utilizes a Disruptor instead of a BlockingQueue. |
| Disruptor VM / camel-disruptor<br>`disruptor-vm:someName[?<option>]` | Same as Disruptor, but the buffers of the Disrupter VM endpoints also provide support for communication across CamelContexts instances so you can use this mechanism to communicate across web applications. |
| SEDA / camel-core<br>`seda:someName[?<options>]` | Asynchronous call to another endpoint in the same Camel Context; staged event-driven architecture (SEDA). |
| VM / camel-core<br>`vm:queueName[?<options>]` | Asynchronous call to another endpoint in the same JVM. |

## ESB

| Component / ArtifactId / URI | Description |
|---|---|
| JBI / servicemix-camel in org.apache.servicemix<br>`jbi:serviceName` | For Java Business Integration (JBI) integration such as working with Apache ServiceMix. |
| NMR / apache-servicemix-nmr in org.apache.servicemix.nmr<br>`nmr://serviceName` | Integration with the Normalized Message Router bus in Apache ServiceMix. |
| Vert.x / camel-vertx<br>`vertx:eventBusName` | Working with the Vert.x event bus. |

## Feeds

| Component / ArtifactId / URI | Description |
|---|---|
| Atom / camel-atom<br>`atom:atomUri[?<options>]` | Working with Apache Abdera for atom integration, such as consuming an atom feed. |
| RSS / camel-rss<br>`rss:uri[?<options>]` | Working with ROME for Rich Site Summary (RSS) integration. |

## File I/O and Transfer

| Component / ArtifactId / URI | Description |
|---|---|
| File / camel-core<br>`file://nameOfFileOrDirectory[?<options>]` | Sending messages to a file or polling a file or directory. |
| Flatpack / camel-flatpack<br>`flatpack:[fixed\|delim]:configFile[?<options>]` | Processing fixed width or delimited files or messages using the FlatPack library. |
| FTP / camel-ftp<br>`ftp:contextPath[?<options>]` | Sending and receiving files over File Transfer Protocol (FTP). |
| FTPS / camel-ftp<br>`ftps://[username@]hostName[:port]/directoryName[?<options>]` | Sending and receiving files over FTP Secure (TLS and SSL). |
| HDFS / camel-hdfs<br>`hdfs://hostName[:port][/path][?<options>]` | For reading/writing from/to an Hadoop Distributed File System (HDFS) filesystem. |
| Jsch / camel-jsch<br>`scp://hostName[:port]/destination[?<options>]` | Support for the scp protocol using the Client API of the Java Secure Channel (JSch) project. |
| SFTP / camel-ftp<br>`ftp://[username@]hostName[:port]/directoryName[?<options>]` | Sending and receiving files over FTP Secure (FTP and SSH). |
| Stream / camel-stream<br>`stream:<in\|out\|err\|file\|header\|url>[?<options>]` | Read or write to an input/output/error/file stream rather like unix pipes. |

## Google

| Component / ArtifactId / URI | Description |
|---|---|
| GAuth / camel-gae<br>`ghttp:contextPath[?<options>]` | Used by web applications to implement an OAuth consumer. |
| GHttp / camel-gae<br>`ghttp:contextPath[?<options>]` | Provides connectivity to the URL fetch service of Google App Engine but can also be used to receive messages from servlets. |
| GLogin / camel-gae<br>`glogin://hostName[:port][?<options>]` | Used by Camel applications outside Google App Engine (GAE) for programmatic login to GAE applications. |
| GTask / camel-gae<br>`gtask://queueName[?<options>]` | Supports asynchronous message processing on Google App Engine by using the task queueing service as message queue. |
| GMail / camel-gae<br>`gmail://user@g[oogle]mail.com[?<options>]` | Supports sending of emails via the mail service of Google App Engine. |
| Guava EventBus / camel-guava-eventbus<br>`guava-eventbus:busName[?<options>]` | The Google Guava EventBus allows publish-subscribe style communication between components without requiring the components to explicitly register with one another. |

## HTTP

| Component / ArtifactId / URI | Description |
|---|---|
| AHC / camel-ahc<br>`ahc:http[s]://hostName[:port][/resourceUri][?<options>]` | To call external HTTP services using Async Http Client. |
| Cometd / camel-cometd<br>`cometd://hostName:port/channelName[?<options>]` | HTTP-based event routing bus used to deliver messages using the Jetty Cometd implementation of the Bayeux Protocol. |
| HTTP / camel-http<br>`http:hostName[:port][/resourceUri][?<options>]` | For calling out to external HTTP servers using Apache HTTP Client 3.x. |
| HTTP4 / camel-http4<br>`http4:hostName[:port][/resourceUri][?<options>]` | For calling out to external HTTP servers using Apache HTTP Client 4.x. |
| Jetty / camel-jetty<br>`jetty:hostName[:port][/resourceUri][?<options>]` | For exposing services over HTTP. |
| Servlet / camel-servlet<br>`servlet:relativePath[?<options>]` | For exposing services over HTTP through the Servlet which is deployed into the Web container. |

## Java Message Service

| Component / ArtifactId / URI | Description |
|---|---|
| ActiveMQ / activemq-camel in org.apache.activemq<br>`activemq:[topic:]destinationName` | For JMS Messaging with Apache ActiveMQ. |
| ActiveMQ Journal / activemq-core<br>`activemq.journal:filesystemDirectory` | Uses ActiveMQ's fast disk journaling implementation to store message bodies in a rolling log file. |
| JMS / camel-jms<br>`jms:[queue:\|topic:]destinationName[?<options>]` | Working with JMS provider. |
| RabbitMQ / camel-rabbitmq<br>`rabbitmq://hostname[:port]/exchangeName[?<options>]` | For JMS Messaging with RabbitMQ. |
| SJMS / camel-sjms<br>`sjms:[queue:\|topic:]destinationName[?<options>]` | From the ground upwards implementation of a JMS client. |
| Stomp / camel-stomp<br>`stomp:queue:destinationName[?<options>]` | For communicating with Stomp compliant message brokers, like Apache ActiveMQ or ActiveMQ Apollo. |
| ZeroMQ / camel-zeromq in camel-extra.<br>`zeromq:<tcp\|ipc>://hostName:port` | For JMS Messaging with ZeroMQ. |

## LDAP

| Component / ArtifactId / URI | Description |
|---|---|
| LDAP / camel-ldap<br>`ldap:hostName[:port][?<options>]` | Performing searches on Lightweight Directory Access Protocol (LDAP) servers. |
| Spring LDAP / camel-spring-ldap<br>`spring-ldap:springLdapTemplateBean[?<options>]` | Camel wrapper for Spring LDAP. Spring LDAP is a Java library for simplifying LDAP operations, based on the pattern of Spring's JdbcTemplate. |

## Mail

| Component / ArtifactId / URI | Description |
|---|---|
| POP3 / camel-mail<br>`pop3://[username@]hostName port][?<options>]` | Receiving email using Post Office Protocol (POP3) and JavaMail. |
| POP3S / camel-mail<br>`pop3s://[username@]hostName port][?<options>]` | Receiving email using secured POP3 and JavaMail. |
| SMTP / camel-mail<br>`smtp://[username@]hostName[:port][?<options>]` | Sending email using Simple Mail Transfer Protocol (SMTP) and JavaMail. |
| SMTPS / camel-mail<br>`smtps://[username@]hostName[:port][?<options>]` | Sending email using secured SMTP and JavaMail. |
| IMAP / camel-mail<br>`imap://[username@]hostName[:port][?<options>]` | Receiving email using Internet Message Access Protocol (IMAP). |
| IMAPS / camel-mail<br>`imaps://[username@]hostName[:port][?<options>]` | Receiving email using secured IMAP. |

## Maintenance and Monitoring

| Component / ArtifactId / URI | Description |
|---|---|
| Browse / camel-core<br>`browse:someName` | Provides a simple BrowsableEndpoint which can be useful for testing, visualisation tools or debugging. The exchanges sent to the endpoint are all available to be browsed. |
| ControlBus / camel-core<br>`controlbus:command[?<options>]` | ControlBus EIP that allows to send messages to Endpoints for managing and monitoring your Camel applications. |
| JMX / camel-jmx<br>`jmx://platform[?<options>]` | For working with Java Management Extensions (JMX) notification listeners. |
| Log / camel-core<br>`log:loggingCategory[?<options>]` | Uses Jakarta Commons Logging to log the message exchange to some underlying logging system like log4j. |
| Nagios / camel-nagios<br>`nagios://hostName[:port][?<options>]` | Sending passive checks to Nagios using JSendNSCA. Nagios supports IT Infrastruture Monitoring. |
| SNMP / camel-snmp<br>`snmp://hostName[:port][?<options>]` | Polling OID values and receiving traps using SNMP via SNMP4J library. |

## Messaging

| Component / ArtifactId / URI | Description |
|---|---|
| AMQP / camel-amqp<br>`amqp:[queue:\|topic:]destinationName[?<options>]` | For Messaging with the Advanced Message Queuing Protocol (AMQP). |
| Esper / camel-esper in camel-extra<br>`esper:name[?<options>]` | Working with the Esper Library for Event Stream Processing. |
| JavaSpace / camel-javaspace<br>`javaspace:jini://hostName[?<options>]` | Sending and receiving messages through JavaSpace. |
| JT/400 / camel-jt400<br>`jt400://user:pwd@system/<path_to_dtaq>[?<options>]` | For integrating with data queues on an AS/400 (i.e., System i, IBM i, i5, etc.) systems. |
| Kestrel / camel-kestrel<br>`kestrel://[addresslist/]queueName[?<options>]` | For producing to or consuming from Kestrel queues. |
| MQTT / camel-mqtt<br>`mqtt:name[?<options>]` | Component for communicating with MQ Telemetry Transport (MQTT) machine-to-machine (M2M) message brokers. |
| SIP / camel-sip<br>`sip://user@hostName[:port][?<options>]` | Publish/Subscribe communication capability using the Session Initiation Protocol (SIP) protocol. |
| SIPS / camel-sip<br>`sips://user@hostName[:port][?<options>]` | Publish/Subscribe communication capability using the secured Session Initiation Protocol (SIP) protocol. |
| SMPP / camel-smpp<br>`smpp://[username@]hostName[:port][?<options>]` | To send and receive SMS using Short Messaging Service Center using the JSMPP library. |
| SMPPS / camel-smpp<br>`smpps://[username@]hostName[:port][?<options>]` | To send and receive secured SMS using Short Messaging Service Center using the JSMPP library. |
| Quickfix / camel-quickfix<br>`quickfix:configFile[?<options>]` | Implementation of the QuickFix for Java engine which allows to send/receive FIX messages. |

## Networking

| Component / ArtifactId / URI | Description |
|---|---|
| DNS / camel-dns<br>`dns:operation[?<options>]` | To lookup domain information and run Domain Name System (DNS) queries using DNSJava. |
| HL7 / camel-hl7<br>`mina2:tcp://hostName[:port][?<options>]` | For working with the HL7 MLLP protocol and the HL7 model using the HAPI library. |
| MINA / camel-mina<br>`mina:<tcp\|udp\|vm>:hostName[:port][?<options>]` | Working with Apache Multipurpose Infrastructure for Network Applications (MINA) 1.x. Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Virtual Machine (VM) protocol are supported. |
| MINA2 / camel-mina2<br>`mina2:<tcp\|udp\|vm>:hostName[:port][?<options>]` | Working with Apache MINA 2.x. Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Virtual Machine (VM) protocol are supported. |
| Netty / camel-netty<br>`netty:<tcp\|udp>//hostName[:port][?<options>]` | Working with TCP and UDP protocols using Java NIO based capabilities offered by the Netty project. |
| Netty HTTP / camel-netty-http<br>`netty-http:http:[port]/contextPath[?<options>]` | Netty HTTP server and client using the Netty project. |

## OSGi

| Component / ArtifactId / URI | Description |
|---|---|
| EventAdmin / camel-eventadmin<br>`eventadmin:topic[?<options>]` | Receiving OSGi EventAdmin events. |
| Pax Logging / camel-paxlogging<br>`paxlogging:appender` | Receiving Pax-Logging events in OSGi. |

## Persistence

| Component / ArtifactId / URI | Description |
|---|---|
| CouchDB / camel-couchdb<br>`couchdb:hostName[:port]/database[?<options>]` | To integrate with Apache CouchDB. |
| Db4o / camel-db4o in camel-extra<br>`db4o://className` | For using a db4o datastore as a queue via the db4o (database for objects) library. |
| EJB / camel-ejb<br>`ejb:ejbName[?<options>]` | Uses the Bean Binding to bind message exchanges to Enterprise Java Beans (EJBs). It works like the Bean component but just for accessing EJBs. Supports EJB 3.0 onwards. |
| HBase / camel-hbase<br>`hbase://table[?<options>]` | For reading/writing from/to an HBase store (Hadoop database). |
| Hibernate / camel-hibernate in camel-extra<br>`hibernate://entityName[?<options>]` | For using a database as a queue via the Hibernate library. |
| iBATIS / camel-ibatis<br>`ibatis://statementName[?<options>]` | Performs a query, poll, insert, update or delete in a relational database using Apache iBATIS. |
| JDBC / camel-jdbc<br>`jdbc:dataSourceName[?<options>]` | For performing Java Database Connectivity (JDBC) queries and operations. |
| JPA / camel-jpa<br>`jpa://entityName[?<options>]` | For using a database as a queue via the Java Persistence API (JPA) specification for working with OpenJPA, Hibernate or TopLink. |
| Krati / camel-krati<br>`krati://datastorePath[?<options>]` | For producing to or consuming to Krati datastores. |
| MongoDB / camel-mongodb<br>`mongodb:connectionBean[?<options>]` | Interacts with MongoDB databases and collections. Offers producer endpoints to perform CRUD-style operations and more against databases and collections, as well as consumer endpoints to listen on collections and dispatch objects to Camel routes. |
| MyBatis / camel-mybatis<br>`mybatis://statementName[?<options>]` | Performs a query, poll, insert, update or delete in a relational database using MyBatis. |
| SQL / camel-sql<br>`sql:queryCommand[?<options>]` | Performing Structured Query Language (SQL) queries using JDBC. |
| Spring Neo4j / camel-spring-neo4j in camel-extra<br>`spring-neo4j:http://hostName[:port]/database[?options]` | Component for producing to Neo4j datastore using the Spring Data Neo4j library. |

## Platform Support

| Component / ArtifactId / URI | Description |
|---|---|
| Activiti / activiti-camel<br>`activiti:camelProcess:serviceTask` | For working with Activiti, a light-weight workflow and Business Process Management (BPM) platform which supports BPMN 2. |
| APNS / camel-apns<br>`apns:<notify\|consumer>[?<options>]` | For sending notifications to Apple iOS devices. |
| Salesforce / camel-salesforce<br>`salesforce:topic[?<options>]` | To integrate with Salesforce. |
| SAP NetWeaver / camel-sap-netweaver<br>`sap-netweaver:hostName[:port][?<options>]` | To integrate with SAP NetWeaver Gateway. |

## Remote Services

| Component / ArtifactId / URI | Description |
|---|---|
| Avro / camel-avro<br>`avro:[transport]:[hostName]:[port][/messageName][?<options>]` | Working with Apache Avro for data serialization. |
| Jclouds / camel-jclouds<br>`jclouds:<blobstore\|compute>:[provider id][?<options>]` | For interacting with cloud compute & blobstore service via Jclouds. |
| JCIFS / camel-jcifs in camel-extra<br>`smb://userName@hostName/shareName[?<options>]` | This component provides access to remote file systems over the CIFS/SMB networking protocol by using the JCIFS library. |
| RMI / camel-rmi<br>`rmi://hostName[:port][?<options>]` | Working with Remote Method Invocation (RMI). |
| SSH component / camel-ssh<br>`ssh:[username[:password]@]hostName[:port][?<options>]` | For sending commands to a SSH server. |

## Search Engines

| Component / ArtifactId / URI | Description |
|---|---|
| ElasticSearch / camel-elasticsearch<br>`elasticsearch://clusterName[?<options>]` | For interfacing with an ElasticSearch server. |
| Lucene / camel-lucene<br>`lucene:searcherName:<insert\|query>[?<options>]` | Uses Apache Lucene to perform Java-based indexing and full text based searches using advanced analysis/tokenization capabilities. |
| Solr / camel-solr<br>`solr://hostName[:port]/solr[?<options>]` | Uses the Solrj client API to interface with an Apache Lucene Solr server. |

## Security

| Component / ArtifactId / URI | Description |
|---|---|
| Crypto (Digital Signatures) / camel-crypto<br>`crypto:<sign\|verify>:name[?<options>]` | Used to sign and verify exchanges using the Signature Service of the Java Cryptographic Extension. |
| XML Security / camel-xmlsecurity<br>`xmlsecurity:<sign\|verify>:name[?<options>]` | Used to sign and verify exchanges using the XML signature specification. |

## Social Media

| Component / ArtifactId / URI | Description |
|---|---|
| Facebook / camel-facebook<br>`facebook://endpoint[?<options>]` | Providing access to all of the Facebook APIs accessible using Facebook4J. |
| Twitter / camel-twitter<br>`twitter://endpoint[?<options>]` | Enables the most useful features of the Twitter API by encapsulating Twitter4J. |
| Yammer / camel-yammer<br>`yammer://function[?<options>]` | Allows you to interact with the Yammer enterprise social network. |

## Special support

| Component / ArtifactId / URI | Description |
|---|---|
| Cache / camel-cache<br>`cache://cacheName[?<options>]` | The cache component facilitates creation of caching endpoints and processors using EHCache as the cache implementation. |
| Geocoder / camel-geocoder<br>`geocoder:<address:\|latlng:latitude,longitude>[?<options>]` | Supports looking up geocoders for an address, or reverse lookup geocoders from an address. |
| Hazelcast / camel-hazelcast<br>`hazelcast://[type]:cachename[?<options>]` | Hazelcast is a data grid supporting map, multimap, SEDA, queue, set, atomic number and simple clusters. |
| RCode / camel-rcode in camel-extra<br>`rcode://host[:port]/operation[?options]` | Uses Rserve to integrate Camel with the statistics environment R. |
| Routebox / camel-routebox<br>`routebox:routeBoxName[?<options>]` | Facilitates the creation of specialized endpoints that offer encapsulation and a strategy/map based indirection service to a collection of camel routes hosted in an automatically created or user injected camel context. |
| Weather / camel-weather<br>`weather://name[?<options>]` | Polls the weather information from Open Weather Map. |

## Spring

| Component / ArtifactId / URI | Description |
|---|---|
| Spring Event / camel-spring<br>`spring-event://default` | Working with Spring Application Events. |
| Spring Batch / camel-spring-batch<br>`spring-batch:jobName[?<options>]` | To bridge Camel and Spring Batch. |
| Spring Integration / camel-spring-integration<br>`spring-integration:defaultChannelName[?<options>]` | The bridge component of Camel and Spring Integration. |
| Spring Redis / camel-spring-redis<br>`spring-redis://hostName:port[?<options>]` | Component for consuming and producing from a Redis key-value store. |

## Templates

| Component / ArtifactId / URI | Description |
|---|---|
| FreeMarker / camel-freemarker<br>`freemarker:templateName[?<options>]` | Generates a response using a FreeMarker template. |
| Mustache / camel-mustache<br>`mustache:templateName[?<options>]` | Generates a response using a Mustache template. |
| MVEL / camel-mvel<br>`mvel:templateName[?<options>]` | Generates a response using an MVFLEX Expression Language (MVEL) template. |
| Scalate / scalate-camel in org.fusesource.scalate<br>`scalate:templateName` | Generates a response using a Scalate template. |
| StringTemplate / camel-stringtemplate<br>`string-template:templateName[?<options>]` | Generates a response using a String Template. |

## Testing

| Component / ArtifactId / URI | Description |
|---|---|
| DataSet / camel-core<br>`dataset:name[?<options>]` | For load and soak testing, the DataSet provides a way to create huge numbers of messages for sending to components or asserting that they are consumed correctly. |
| Mock / camel-core<br>`mock:name[?<options>]` | For testing routes and mediation rules using mocks. |
| Stub / camel-core<br>`stub:someOtherCamelUri[?<options>]` | Allows you to stub out some physical middleware endpoint for easier testing or debugging. |
| Test / camel-spring<br>`test:expectedMessagesEndpointUri[?<options>]` | Creates a Mock endpoint which expects to receive all the message bodies that could be polled from the given underlying endpoint. |

## Web Services

| Component / ArtifactId / URI | Description |
|---|---|
| CXF / camel-cxf<br>`cxf:<bean:cxfEndpoint\|//someAddress>[?<options>]` | Working with Apache CXF for web services integration. |
| CXF Bean / camel-cxf<br>`cxfbean:serviceBeanRef[?<options>]` | Proceess the exchange using a JAX-WS or JAX-RS annotated bean from the registry. |
| CXFRS / camel-cxf<br>`cxfrs:<bean:rsEndpoint\|//someAddress>[?<options>]` | Working with Apache CXF for RESTful services integration. |
| Restlet / camel-restlet<br>`restlet:restletUri[?<options>]` | Component for consuming and producing Restful resources using Restlet. |
| Spring Web Services / camel-spring-ws<br>`spring-ws:[mapping-type:]someAddress[?<options>]` | Client-side support for accessing web services, and server-side support for creating your own contract-first web services using Spring Web Services. |
| Websocket / camel-websocket<br>`websocket://hostname[:port][/resourceUri][?<options>]` | Communicating with Websocket clients. |

## XML

| Component / ArtifactId / URI | Description |
|---|---|
| Bean Validation / camel-bean-validator<br>`bean-validator:something[?<options>]` | Validates the payload of a message using the Java Validation API (JSR 303). Hibernate Validator is the reference implementation. |
| FOP / camel-fop<br>`fop:outputFormat[?<options>]` | Renders the message into different output formats using Formatting Objects Processor (FOP), which is driven by XSL formatting objects (XSL-FO). |
| MSV / camel-msv<br>`msv:someLocalOrRemoteResource[?<options>]` | Validates the payload of a message using the MSV Library. |
| RNC / camel-jing<br>`rnc:/relativeOrAbsoluteUri[?<options>]` | Validates the payload of a message using RelaxNG Compact Syntax. |
| RNG / camel-jing<br>`rng:/relativeOrAbsoluteUri[?<options>]` | Validates the payload of a message using RelaxNG. |
| StAX / camel-stax<br>`stax:(contentHandlerClassName\|#myHandler)` | Process messages through a Simple API for XML (SAX) ContentHandler. |
| XQuery / camel-saxon<br>`xquery:someXQueryResource` | Generates a response using an XQuery template. |
| XSLT / camel-core<br>`xslt:templateName[?<options>]` | Generates a response using a Extensible Stylesheet Language Transformations (XSLT) template. |
| Validation / camel-core<br>`validation:someLocalOrRemoteResource[?<options>]` | Validates the payload of a message using XML Schema and Java API for XML Processing (JAXP) Validation. |

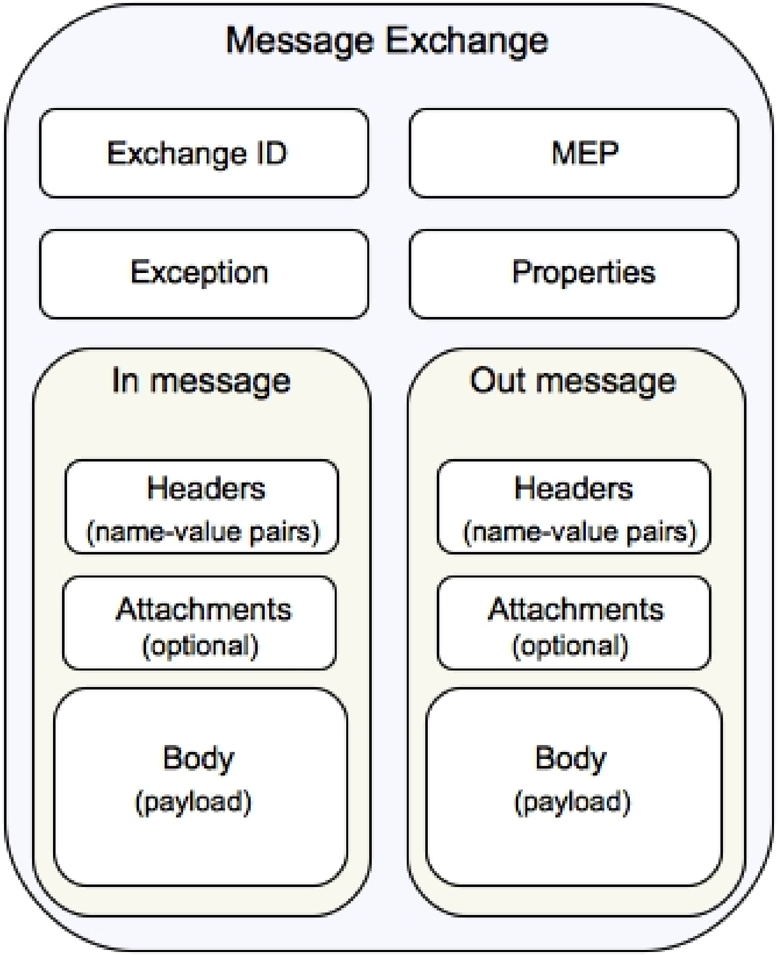
Camel Components (Apache and External)

Gliesian, LLC

http://camel.apache.org/components.html

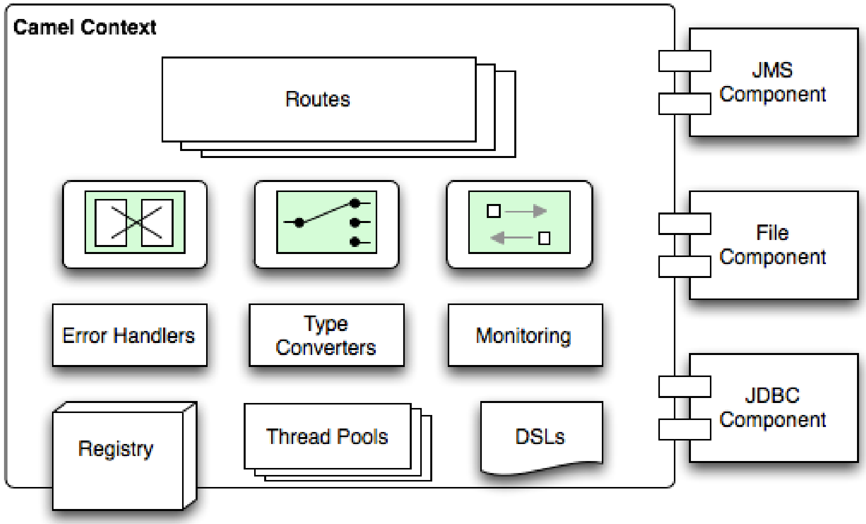
# Message and Exchange

- Messages wrap requests, responses and exceptions and contain
  - Headers
  - Payload
  - Attachments
- Exchange
  - Created by a producer
  - Consumed by a consumer
  - Contains
    - Unique ID
    - Incoming and outgoing message
    - Exception
    - Properties



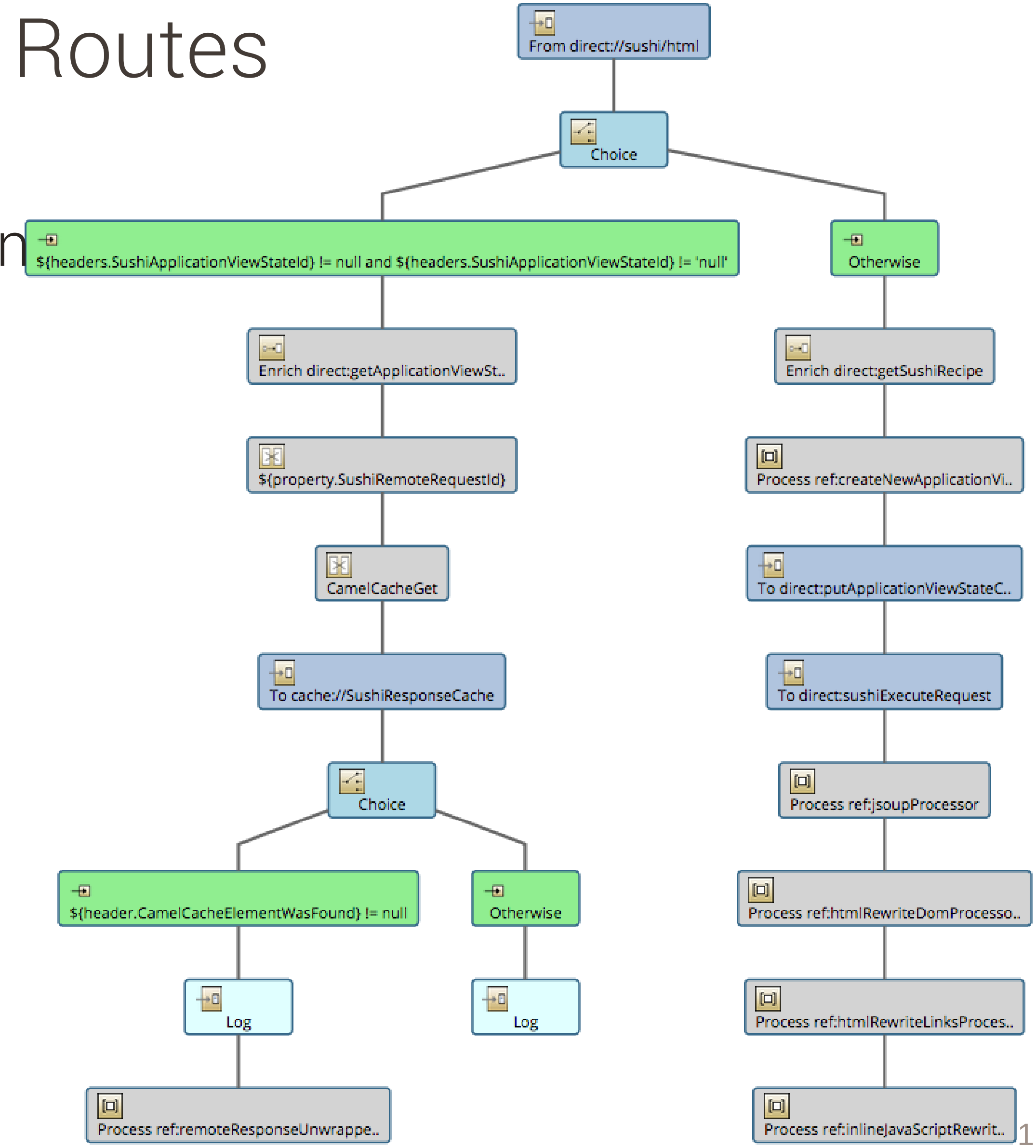
# Camel Context

- Typically one per application
- Provides the runtime for the camel routes
- Serves as a registry for endpoints and components
- Camel Contexts contains Routes and Route Contexts
- Routes can be grouped together in Route Contexts



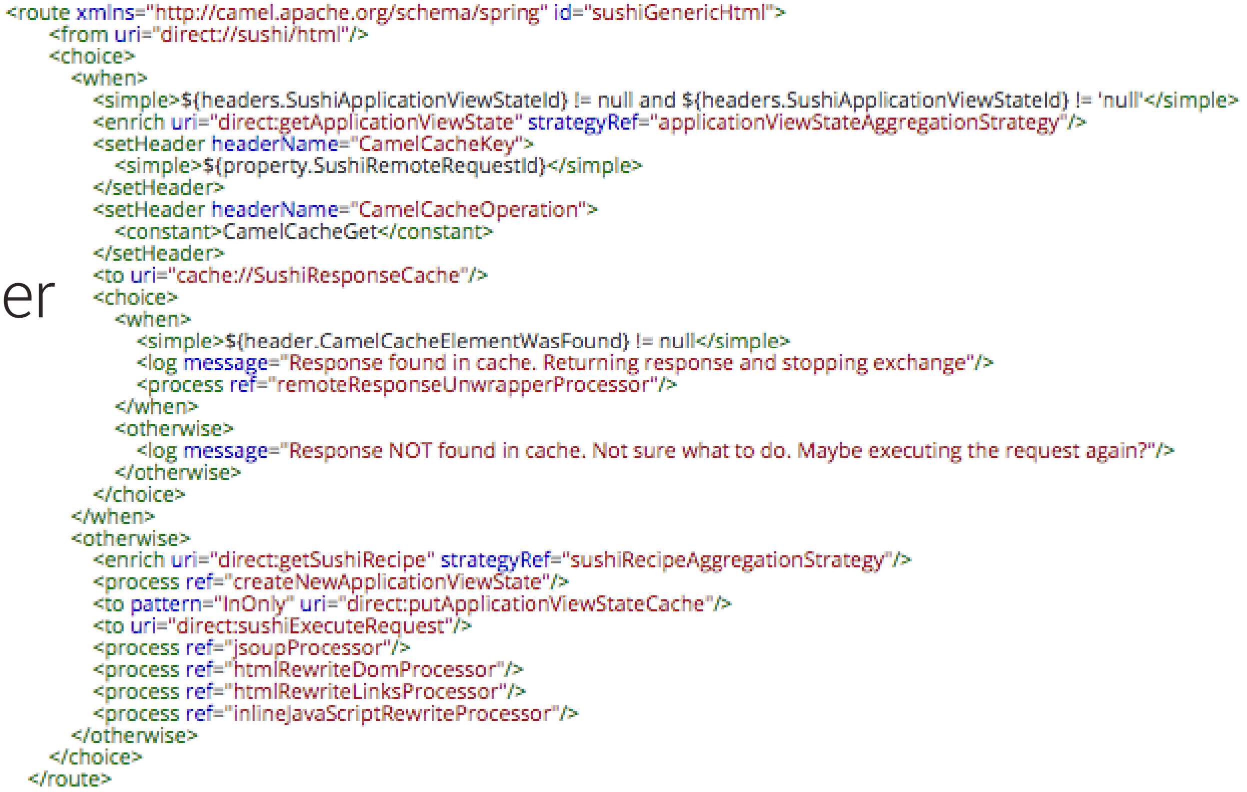
# Routes

- Step-by-step movement of a message through an exchange.
- Configured by DSL
  - Spring XML
  - Java DSL


From direct://sushi/html
Choice
${headers.SushiApplicationViewStateId} != null and ${headers.SushiApplicationViewStateId} != 'null'
Otherwise
Enrich direct:getApplicationViewSt..
Enrich direct:getSushiRecipe
${property.SushiRemoteRequestId}
Process ref:createNewApplicationVi..
CamelCacheGet
To direct:putApplicationViewStateC..
To cache://SushiResponseCache
To direct:sushiExecuteRequest
Choice
Process ref:jsoupProcessor
${header.CamelCacheElementWasFound} != null
Otherwise
Process ref:htmlRewriteDomProcesso..
Log
Log
Process ref:htmlRewriteLinksProces..
Process ref:remoteResponseUnwrappe..
Process ref:inlineJavaScriptRewrit..

# Spring XML

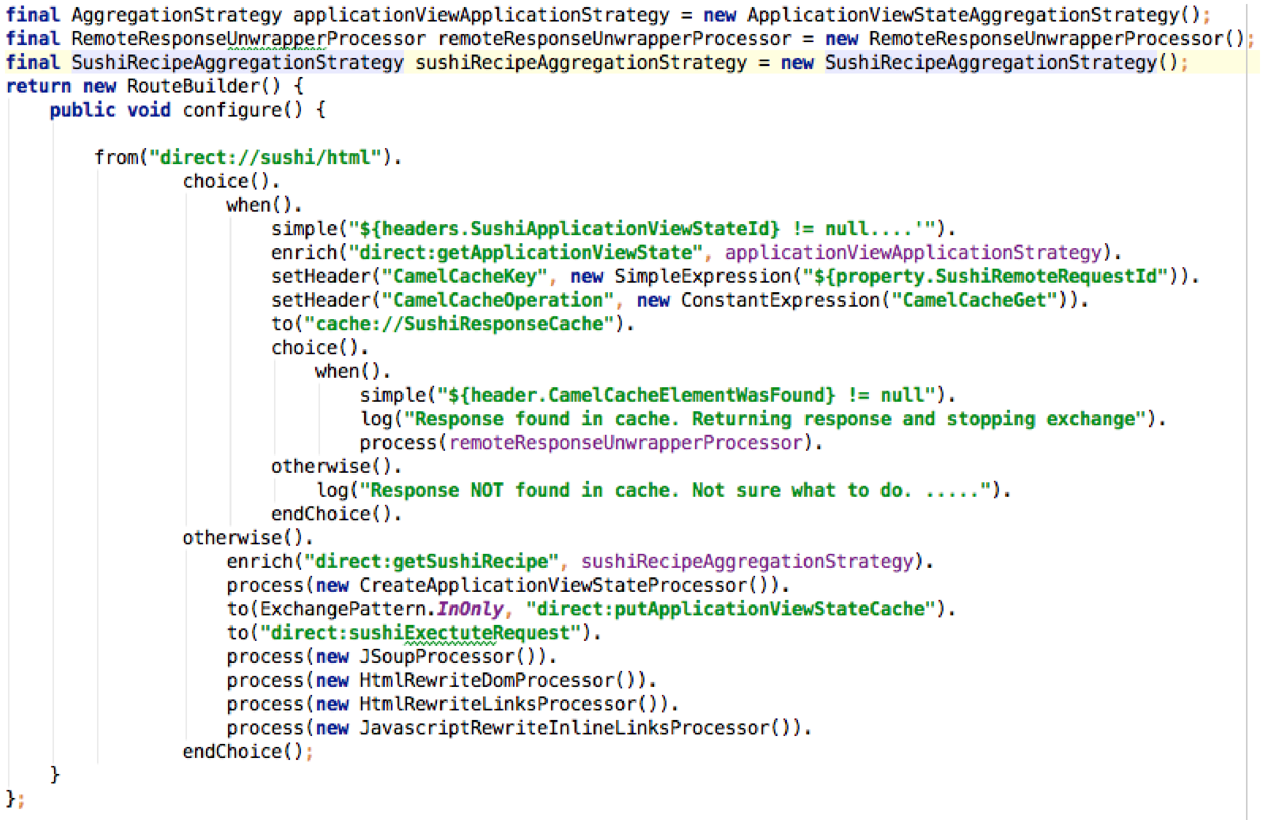
- Uses Camel XSD Schema
- Verbose
- Property Placeholder Support



```
<route xmlns="http://camel.apache.org/schema/spring" id="sushiGenericHtml">
    <from uri="direct://sushi/html"/>
    <choice>
      <when>
        <simple>${headers.SushiApplicationViewStateId} != null and ${headers.SushiApplicationViewStateId} != 'null'</simple>
        <enrich uri="direct:getApplicationViewState" strategyRef="applicationViewStateAggregationStrategy"/>
        <setHeader headerName="CamelCacheKey">
          <simple>${property.SushiRemoteRequestId}</simple>
        </setHeader>
        <setHeader headerName="CamelCacheOperation">
          <constant>CamelCacheGet</constant>
        </setHeader>
        <to uri="cache://SushiResponseCache"/>
        <choice>
          <when>
            <simple>${header.CamelCacheElementWasFound} != null</simple>
            <log message="Response found in cache. Returning response and stopping exchange"/>
            <process ref="remoteResponseUnwrapperProcessor"/>
          </when>
          <otherwise>
            <log message="Response NOT found in cache. Not sure what to do. Maybe executing the request again?"/>
          </otherwise>
        </choice>
      </when>
      <otherwise>
        <enrich uri="direct:getSushiRecipe" strategyRef="sushiRecipeAggregationStrategy"/>
        <process ref="createNewApplicationViewState"/>
        <to pattern="InOnly" uri="direct:putApplicationViewStateCache"/>
        <to uri="direct:sushiExecuteRequest"/>
        <process ref="jsoupProcessor"/>
        <process ref="htmlRewriteDomProcessor"/>
        <process ref="htmlRewriteLinksProcessor"/>
        <process ref="inlineJavaScriptRewriteProcessor"/>
      </otherwise>
    </choice>
  </route>
```

# Java DSL

- Syntactic sugar
- Difficult to format
- Autocomplete awareness in IDE



```
final AggregationStrategy applicationViewApplicationStrategy = new ApplicationViewStateAggregationStrategy();
final RemoteResponseUnwrapperProcessor remoteResponseUnwrapperProcessor = new RemoteResponseUnwrapperProcessor();
final SushiRecipeAggregationStrategy sushiRecipeAggregationStrategy = new SushiRecipeAggregationStrategy();
return new RouteBuilder() {
    public void configure() {

        from("direct://sushi/html").
                choice().
                    when().
                        simple("${headers.SushiApplicationViewStateId} != null....'").
                        enrich("direct:getApplicationViewState", applicationViewApplicationStrategy).
                        setHeader("CamelCacheKey", new SimpleExpression("${property.SushiRemoteRequestId")).
                        setHeader("CamelCacheOperation", new ConstantExpression("CamelCacheGet")).
                        to("cache://SushiResponseCache").
                        choice().
                            when().
                                simple("${header.CamelCacheElementWasFound} != null").
                                log("Response found in cache. Returning response and stopping exchange").
                                process(remoteResponseUnwrapperProcessor).
                        otherwise().
                            log("Response NOT found in cache. Not sure what to do. .....").
                        endChoice().
                otherwise().
                    enrich("direct:getSushiRecipe", sushiRecipeAggregationStrategy).
                    process(new CreateApplicationViewStateProcessor()).
                    to(ExchangePattern.InOnly, "direct:putApplicationViewStateCache").
                    to("direct:sushiExectuteRequest").
                    process(new JSoupProcessor()).
                    process(new HtmlRewriteDomProcessor()).
                    process(new HtmlRewriteLinksProcessor()).
                    process(new JavascriptRewriteInlineLinksProcessor()).
                endChoice();
    }
};
```

# Other DSL’s

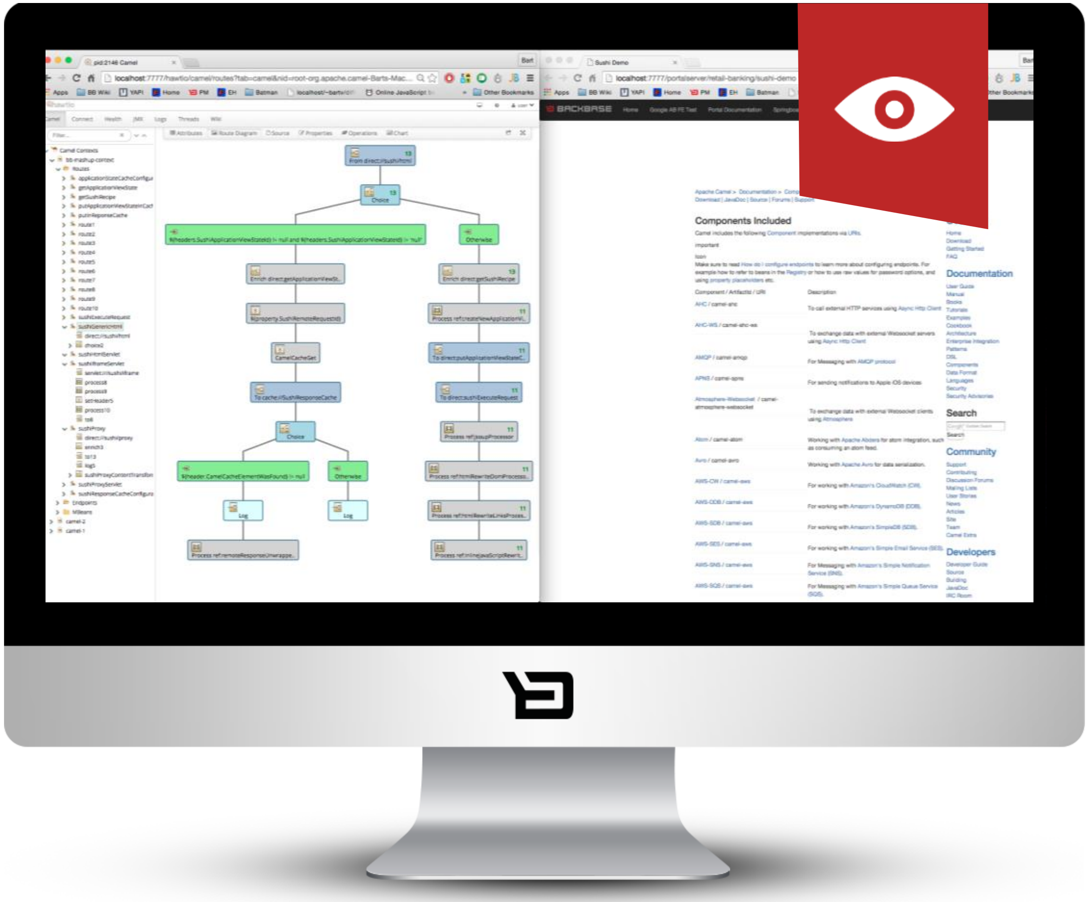
- Blueprint XML - A XML based DSL in OSGi Blueprint XML files
- Rest DSL - A DSL to define REST services using a REST style in either Java or XML.
- Groovy DSL - A Groovy based DSL using Groovy programming language
- Scala DSL - A Scala based DSL using Scala programming language
- Annotation DSL - Use annotations in Java beans.

# Processors

```
public class JSoupProcessor implements Processor {


    @Override
    public void process(Exchange exchange) throws Exception {
        String currentBaseUri = exchange.getIn().getHeader(Exchange.HTTP_URI, String.class);
        // Retrieve Input Stream from Body
        InputStream inputStream = exchange.getIn().getBody(InputStream.class);

        String contentType = exchange.getIn().getHeader("Content-Type",String.class);
        String currentEncoding = "UTF-8";
        if(contentType.contains(";")) {
            String[] parts = contentType.split(";");
            for(String part : parts) {
                if(part.startsWith("charset")) {
                    currentEncoding = StringUtils.substringAfter(part,"=");
                    // Set Content Type to UTF-8
                    contentType = contentType.replace(currentEncoding, "UTF-8");
                    exchange.getIn().setHeader(Exchange.CONTENT_TYPE, contentType);
                }
            }
        }

        String body = IOUtils.toString(inputStream,currentEncoding);
        String encodedBody = new String(body.getBytes(),"UTF-8");
        // Parse input stream with JSoup
        Document document = Jsoup.parse(encodedBody, currentBaseUri);
        // Store parsed document in the OUT
        exchange.getIn().setBody(document);
        // Gzipped encoded pages are automatically deflated.
        // Make sure to remove the header so contents are not deflated twice
        exchange.getIn().removeHeader(Exchange.CONTENT_ENCODING);
    }
}
```

- Processes an exchange
- Can modify the incoming message
- Or set a new outgoing message
- Possibilities are endless

# Apache Camel


localhost:7777/hawtio/camel/routes?tab=camel&nid=root-org.apache.camel-Barts-Mac...
Attributes
Route Diagram
Source
Properties
Operations
Chart
From direct://sushihtml
Choice
${headers.SushiApplicationViewStateId} != null and ${headers.SushiApplicationViewStateId} != 'null'
Otherwise
Enrich direct:getApplicationViewSt...
${property.SushiRemoteRequestId}
CamelCacheGet
To cache://SushiResponseCache
Choice
${header.CamelCacheElementWasFound} != null
Otherwise
Log
Log
Process ref:remoteResponseUnwrappe...
Enrich direct:getSushiRecipe
Process ref:createNewApplicationVi...
To direct:putApplicationViewStateC...
To direct:sushiExecuteRequest
Process ref:jsoupProcessor
Process ref:htmlRewriteDomProcess...
Process ref:htmlRewriteLinksProcess...
Process ref:inlineJavaScriptRewrit...
localhost:7777/portalserver/retail-banking/sushi-demo
Home
Google AB FE Test
Portal Documentation

Apache Camel > Documentation > Comp

Download | JavaDoc | Source | Forums | Support

## Components Included

Camel includes the following Component implementations via URIs.

important

Icon

Make sure to read How do I configure endpoints to learn more about configuring endpoints. For example how to refer to beans in the Registry or how to use raw values for password options, and using property placeholders etc.

| Component / Artifactid / URI | Description |
|---|---|
| AHC / camel-ahc | To call external HTTP services using Async Http Client |
| AHC-WS / camel-ahc-ws | To exchange data with external Websocket servers using Async Http Client |
| AMQP / camel-amqp | For Messaging with AMQP protocol |
| APNS / camel-apns | For sending notifications to Apple iOS devices |
| Atmosphere-Websocket / camel-atmosphere-websocket | To exchange data with external Websocket clients using Atmosphere |
| Atom / camel-atom | Working with Apache Abdera for atom integration, such as consuming an atom feed. |
| Avro / camel-avro | Working with Apache Avro for data serialization. |
| AWS-CW / camel-aws | For working with Amazon's CloudWatch (CW). |
| AWS-DDB / camel-aws | For working with Amazon's DynamoDB (DDB). |
| AWS-SDB / camel-aws | For working with Amazon's SimpleDB (SDB). |
| AWS-SES / camel-aws | For working with Amazon's Simple Email Service (SES). |
| AWS-SNS / camel-aws | For Messaging with Amazon's Simple Notification Service (SNS). |
| AWS-SQS / camel-aws | For Messaging with Amazon's Simple Queue Service (SQS). |

Home
Download
Getting Started
FAQ

Documentation

User Guide
Manual
Books
Tutorials
Examples
Cookbook
Architecture
Enterprise Integration Patterns
DSL
Components
Data Format
Languages
Security
Security Advisories

Search

Search

Community

Support
Contributing
Discussion Forums
Mailing Lists
User Stories
News
Articles
Site
Team
Camel Extra

Developers

Developer Guide
Source
Building
JavaDoc
IRC Room

# Break

# Apache Camel In CXP

How to work with Apache Camel and CXP

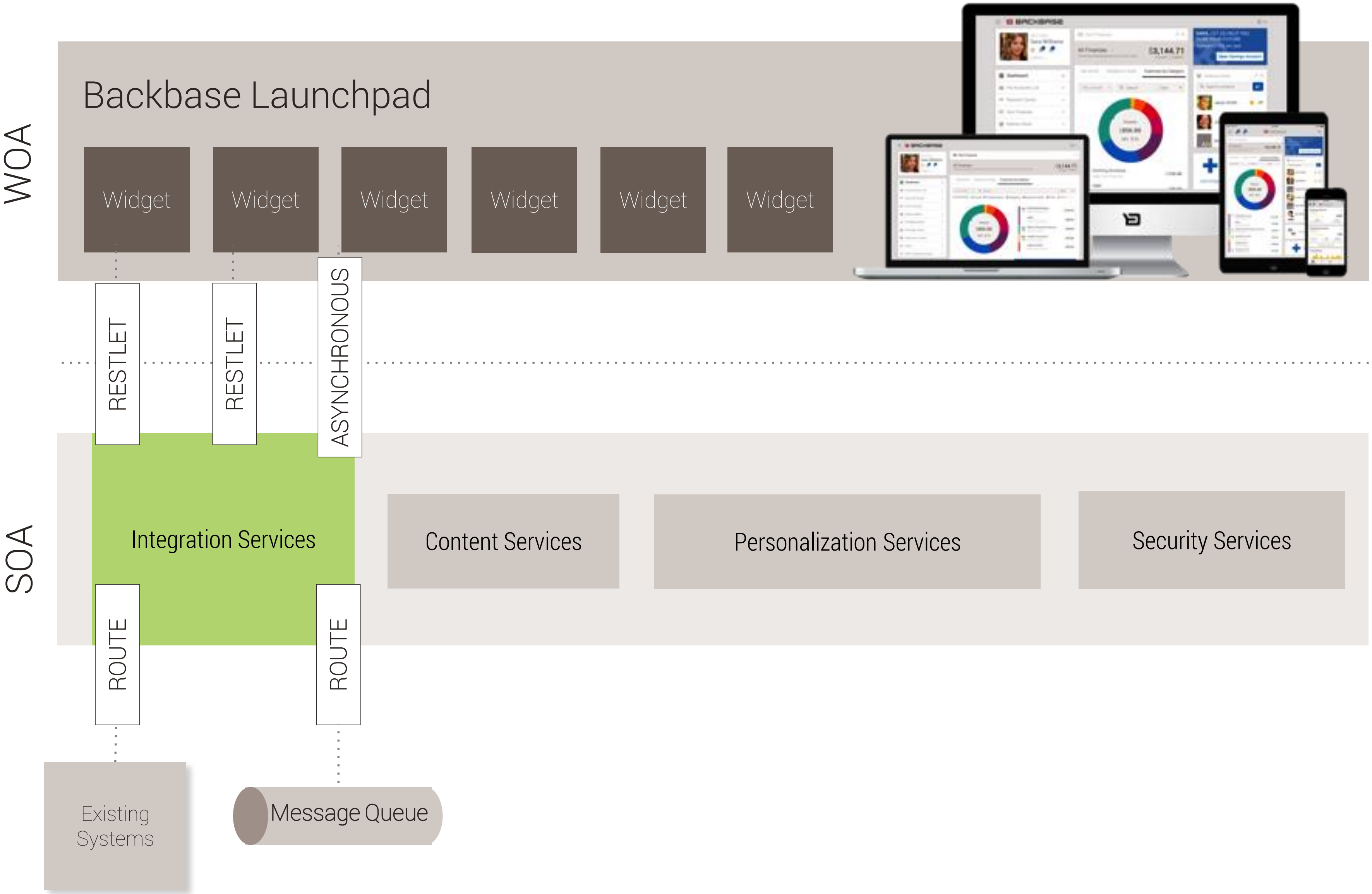
- Overview
- Architecture
- Implementation details
- Exercises
    - Setup Camel Module
    - Data Integration
        - REST
        - Web Services (SOAP)
        - Active MQ (optional)
    - HTML Integration
        - Sushi (build simple recipe)
        - g:include
    - Camel Servlet Component
        - Opaque services

# Apache Camel In CXP

- Integrated in Launchpad CXP
- Apache Camel 2.12.4
- To remain compatible with JBoss Fuse for commercial support
- Camel Context defined by Launchpad
    - Defines default RESTlet $(contextRoot)/services/rest/*
    - ResletRouterProcessor forwards requests to RESTlet routes
- Route Contexts are loaded automatically from spring context files
    - classpath*:META-INF/spring/backbase-integration-service.xml Route Context IDs must be prefixed with
        - com.backbase.portal.integration.service
- Embedded or stand alone

BACKBASE

# Apache Camel In CXP



WOA

Backbase Launchpad

Widget

Widget

Widget

Widget

Widget

Widget

RESTLET

RESTLET

ASYNCHRONOUS

SOA

Integration Services

Content Services

Personalization Services

Security Services

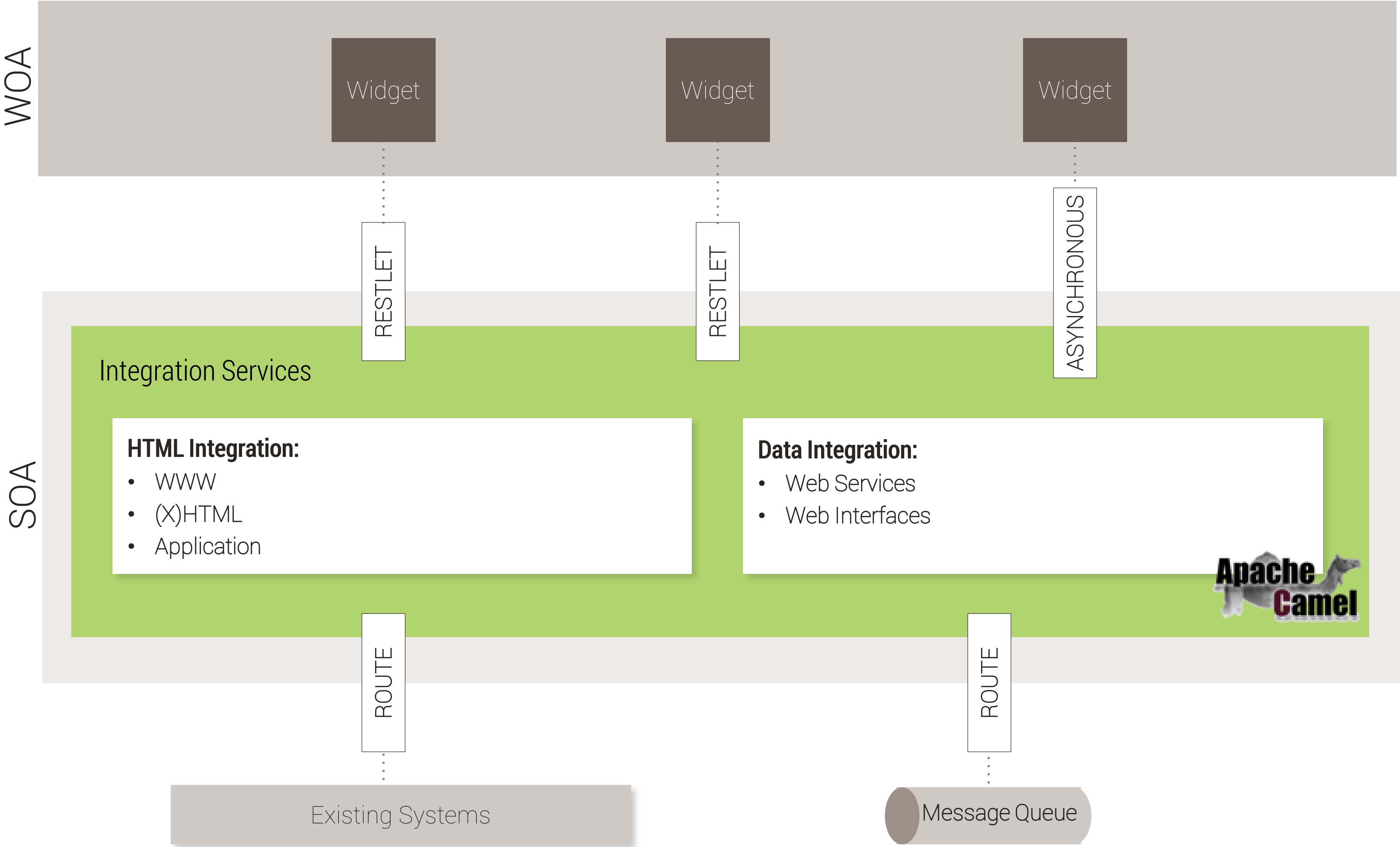
ROUTE

ROUTE

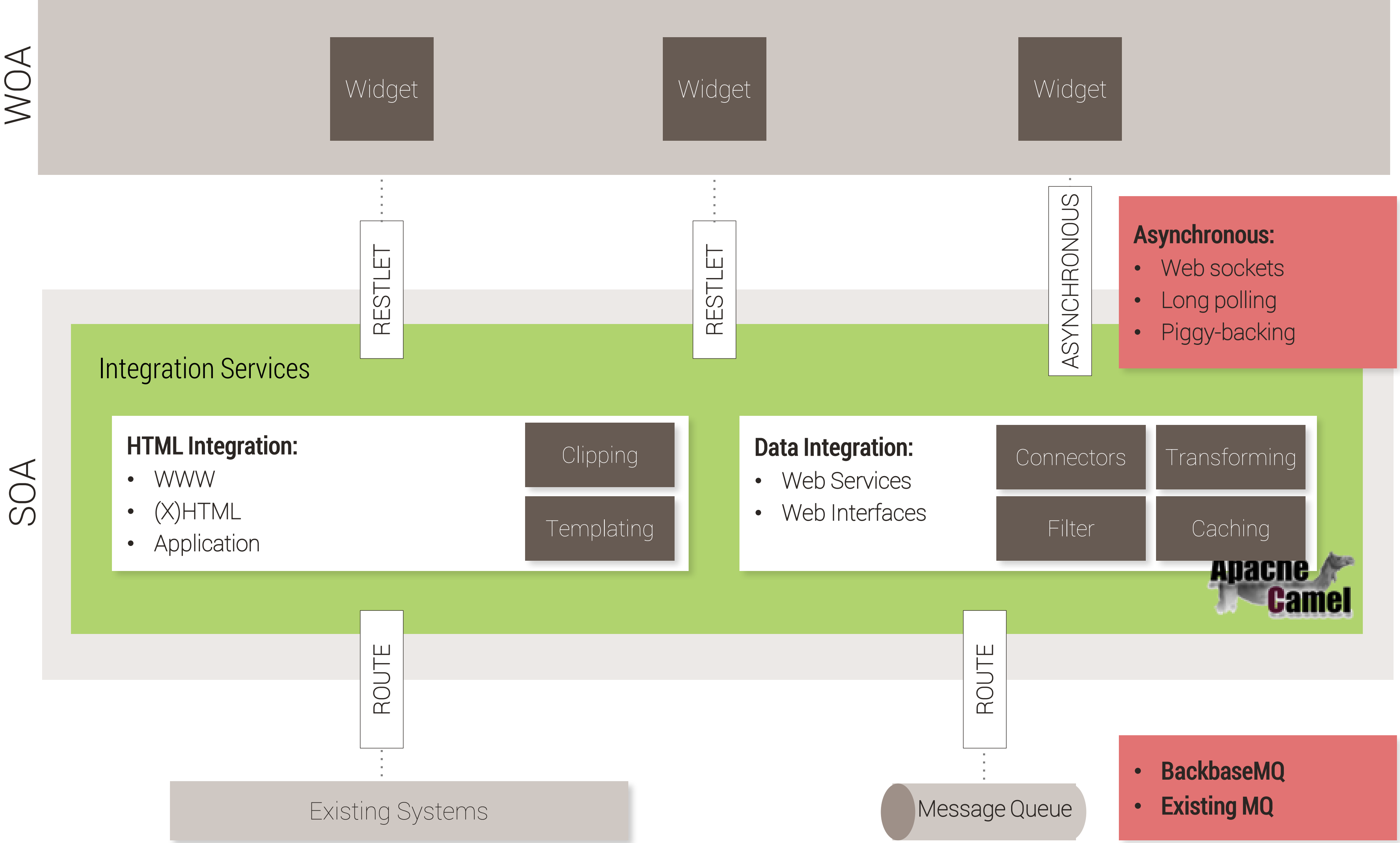
Existing Systems

Message Queue

# Apache Camel In CXP



# Apache Camel In CXP



WOA

Widget

Widget

Widget

RESTLET

RESTLET

ASYNCHRONOUS

**Asynchronous:**

- Web sockets
- Long polling
- Piggy-backing

SOA

Integration Services

**HTML Integration:**

- WWW
- (X)HTML
- Application

Clipping

Templating

**Data Integration:**

- Web Services
- Web Interfaces

Connectors

Transforming

Filter

Caching

Apache Camel

ROUTE

ROUTE

Existing Systems

Message Queue

- **BackbaseMQ**
- **Existing MQ**

# Implementation details

- Camel Context is defined in backbase-integration.xml
- backbase-integration.xml is added to contextConfigurationLocation array in web.xml

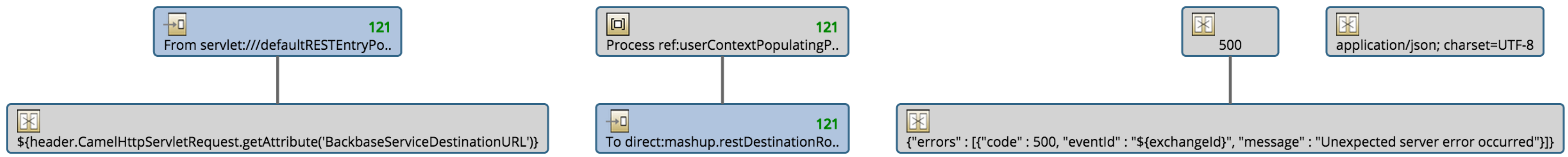
```
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value><![CDATA[
        classpath:/META-INF/spring/backbase-portal-application-config.xml
        classpath:/META-INF/spring/backbase-portal-integration-config.xml
        classpath:/META-INF/spring/backbase-portal-presentation-security.xml
        classpath:/META-INF/spring/backbase-portal-presentation-ssr.xml
        classpath:/META-INF/spring/backbase-portal-business-security.xml
        classpath:backbase-ptc.xml
        classpath:/META-INF/spring/backbase-integration.xml
    ]]></param-value>
</context-param>
```

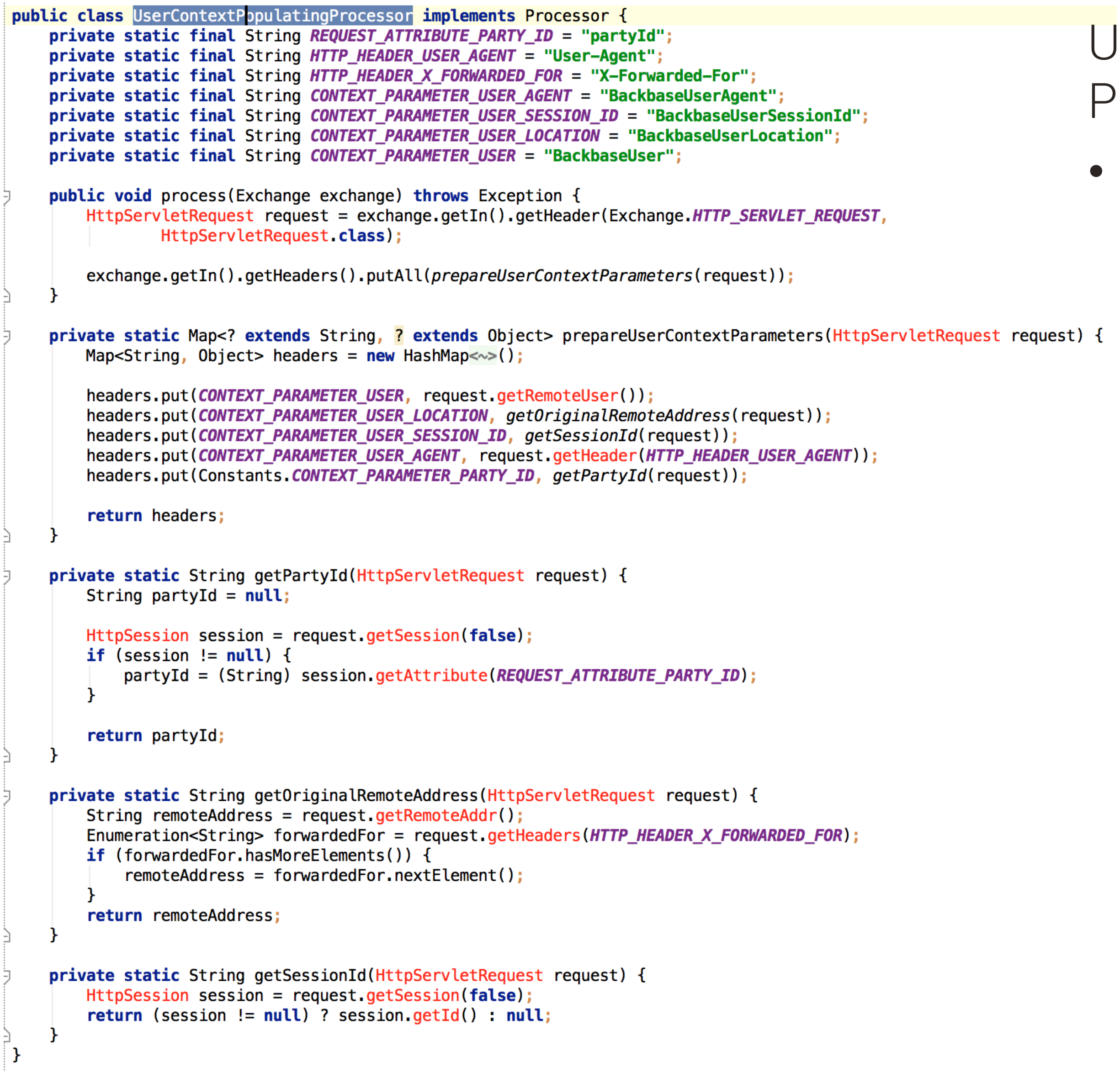
- Integration Servlet in web.xml defines main entry point

```
<servlet>
    <servlet-name>Integration Servlet</servlet-name>
    <servlet-class>com.backbase.portal.integration.core.servlet.IntegrationServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
    <servlet-name>Integration Servlet</servlet-name>
    <url-pattern>/services/*</url-pattern>
</servlet-mapping>
```

# Default REST entry point

```
<route xmlns="http://camel.apache.org/schema/spring" id="route4">
    <from uri="servlet:///defaultRESTEntryPoint?matchOnUriPrefix=true&amp;headerFilterStrategy=#defaultHeaderFilterStrategy&amp;httpBindingRef=#defaultHttpBinding"/>
    <setHeader headerName="BackbaseServiceDestinationURL">
      <simple>${header.CamelHttpServletRequest.getAttribute('BackbaseServiceDestinationURL')}</simple>
    </setHeader>
    <doTry>
      <process ref="userContextPopulatingProcessor"/>
      <to uri="direct:mashup.restDestinationRoute"/>
      <doCatch>
        <exception>java.lang.Exception</exception>
        <setHeader headerName="CamelHttpResponseCode">
          <constant>500</constant>
        </setHeader>
        <setBody>
          <simple>
              {"errors" : [{"code" : 500, "eventId" : "${exchangeId}", "message" : "Unexpected server error occurred"}]}
            </simple>
        </setBody>
      </doCatch>
      <doFinally>
        <setHeader headerName="Content-Type">
          <simple>application/json; charset=UTF-8</simple>
        </setHeader>
      </doFinally>
    </doTry>
  </route>
```


121
From servlet:///defaultRESTEntryPo..
${header.CamelHttpServletRequest.getAttribute('BackbaseServiceDestinationURL')}
121
Process ref:userContextPopulatingP..
121
To direct:mashup.restDestinationRo..
500
application/json; charset=UTF-8
{"errors" : [{"code" : 500, "eventId" : "${exchangeId}", "message" : "Unexpected server error occurred"}]}



```java
public class UserContextPopulatingProcessor implements Processor {
    private static final String REQUEST_ATTRIBUTE_PARTY_ID = "partyId";
    private static final String HTTP_HEADER_USER_AGENT = "User-Agent";
    private static final String HTTP_HEADER_X_FORWARDED_FOR = "X-Forwarded-For";
    private static final String CONTEXT_PARAMETER_USER_AGENT = "BackbaseUserAgent";
    private static final String CONTEXT_PARAMETER_USER_SESSION_ID = "BackbaseUserSessionId";
    private static final String CONTEXT_PARAMETER_USER_LOCATION = "BackbaseUserLocation";
    private static final String CONTEXT_PARAMETER_USER = "BackbaseUser";

    public void process(Exchange exchange) throws Exception {
        HttpServletRequest request = exchange.getIn().getHeader(Exchange.HTTP_SERVLET_REQUEST,
                HttpServletRequest.class);

        exchange.getIn().getHeaders().putAll(prepareUserContextParameters(request));
    }

    private static Map<? extends String, ? extends Object> prepareUserContextParameters(HttpServletRequest request) {
        Map<String, Object> headers = new HashMap<~>();

        headers.put(CONTEXT_PARAMETER_USER, request.getRemoteUser());
        headers.put(CONTEXT_PARAMETER_USER_LOCATION, getOriginalRemoteAddress(request));
        headers.put(CONTEXT_PARAMETER_USER_SESSION_ID, getSessionId(request));
        headers.put(CONTEXT_PARAMETER_USER_AGENT, request.getHeader(HTTP_HEADER_USER_AGENT));
        headers.put(Constants.CONTEXT_PARAMETER_PARTY_ID, getPartyId(request));

        return headers;
    }

    private static String getPartyId(HttpServletRequest request) {
        String partyId = null;

        HttpSession session = request.getSession(false);
        if (session != null) {
            partyId = (String) session.getAttribute(REQUEST_ATTRIBUTE_PARTY_ID);
        }

        return partyId;
    }

    private static String getOriginalRemoteAddress(HttpServletRequest request) {
        String remoteAddress = request.getRemoteAddr();
        Enumeration<String> forwardedFor = request.getHeaders(HTTP_HEADER_X_FORWARDED_FOR);
        if (forwardedFor.hasMoreElements()) {
            remoteAddress = forwardedFor.nextElement();
        }
        return remoteAddress;
    }

    private static String getSessionId(HttpServletRequest request) {
        HttpSession session = request.getSession(false);
        return (session != null) ? session.getId() : null;
    }
}
```

# User Context Populating Processor

- Populates information about logged in user

# Considerations

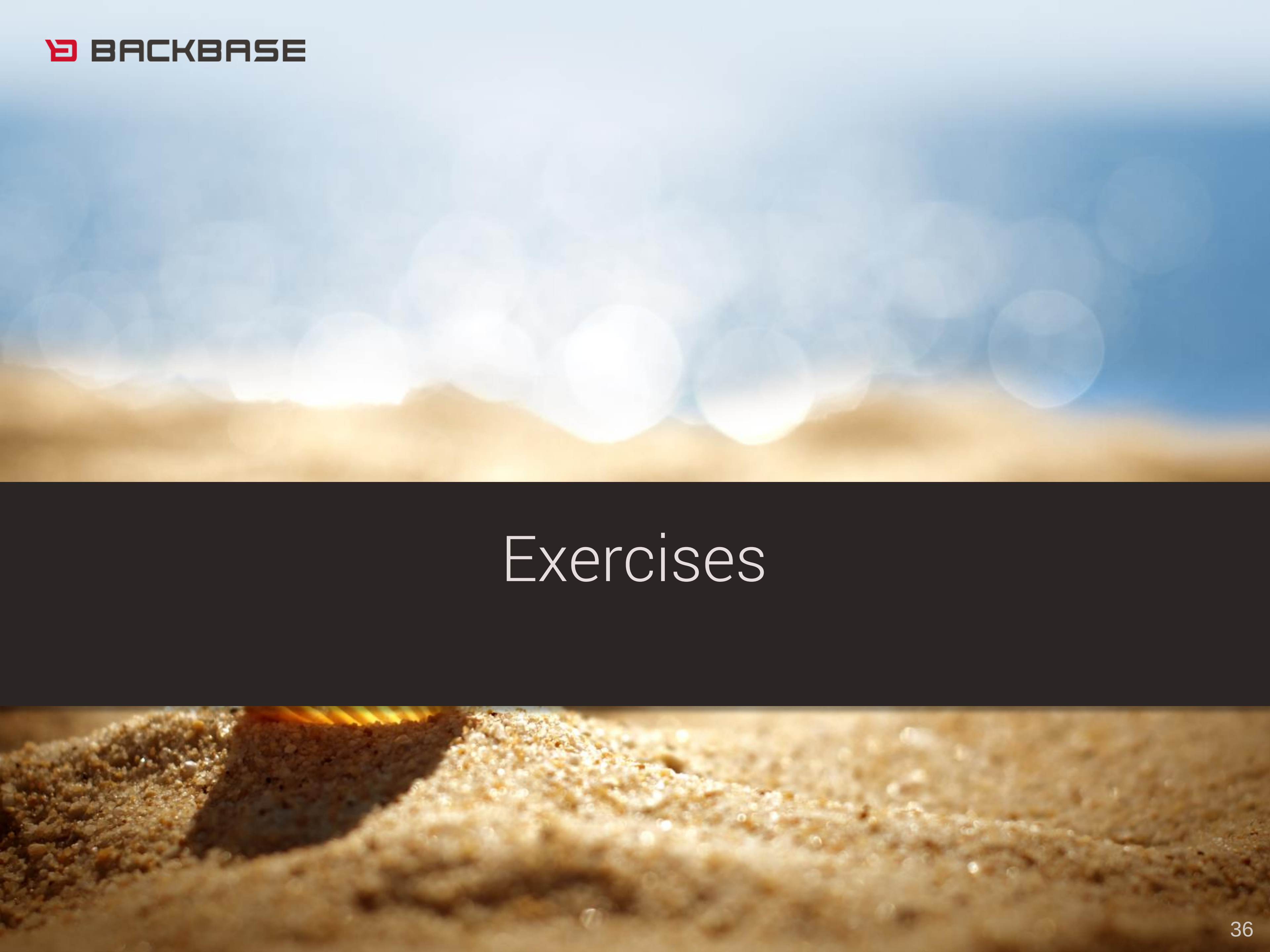
- Launchpad REST services are JSON only.
- All exceptions are caught and are not forwarded to FE for security reasons
- Route Contexts will only be loaded when prefixed with com.backbase.portal.integration.service

# Non JSON Services

- XML
- File Upload
- File Download
- Anything non JSON

- Servlet component puts the raw request on the exchange
  - Must set servletName to Integration Servlet, otherwise Integration servlet cannot find the Servlet component
-
  ```
  <from uri="servlet:///sushi/html?matchOnUriPrefix=true&amp;httpBindingRef=#sushiHttpBinding&amp;servletName=Integration
  Servlet"/>
  ```
  - sushi/html is now mapped on $(contextRoot)/services/sushi/html*

# Break



# Exercises

# Preparations

Load exercise environment in IntelliJ Community Edition

Add training-server and enterprise-integration-widgets maven projects to project

- Add resourceBase to jetty plugin configuration in portalserver/pom.xml pointing to enterprise-integration-widgets/src/main/webapp

Change Maven runner JVM arguments

- -Xmx512m –XX:MaxPermSize=256m

Change log levels to show debug messages

Create run configurations

- Portal Server | jetty:run
- Orchestrator | jetty:run
- Content Services | jetty:run
- Training Server | jetty:run

Start all projects

Add training widgets to catalog using YAPI

# Exercise Environment

- Training Server
  - Manages Players and High Scores
  - Endpoint for
    - REST Services
    - Web Services
    - Active MQ
    - Embedded LDAP server (not connected to player services)
- Start with
  - mvn jetty:run
- Located in training-server

# Exercise: Setup your first Camel Module

- Create maven module in services module
    - Copy sample-service-module
    - pom.xml
        - Change artifcatId
    - backbase-integration-service.xml
        - Change routeContext id
- Add module to services modules
- Add new maven module to portalserver dependencies
- Quick tip
    - Change portal jetty:run configuration to run maven goal “clean install” of services before launching jetty:run to ensure you always have the latest modules
    - Optional*: Enable use of backbase.properties in URL
        - Add dependency
            - com.backbase.launchpad.services.foundation:demo-services-integration:0.11.2

# Exercise: Create Simple Restlets

- Create Player Registration REST service
    - Define route in backbase-integration-service.xml
        - Create a producer using the restlet component
            - Set Method to POST
            - Set Restlet Binding to queryStringToHeadersRestletBinding
        - Log incoming message using the log component
        - Create a consumer that forwards the request to the training-server REST service
            - Use the HTTP component
            - Remove CamelHttp* headers to avoid conflicts (exclude CamelHttpMethod)
                - » Try without this to find out why
        - Log outgoing message
- Test in REST client

http://localhost:9999/training-server/rest?_wadl
http://camel.apache.org/restlet.html
http://camel.apache.org/log.html

# Exercise: Create Simple Player List Restlet

- Define a route to list players
    - Set method to GET
- Transform player list
    - Sort on player attribute with sort query string parameter
- Steps
    - Add camel-jackson dependency
    - Unmarshall JSON into Java Objects
        - Use Jackson library
    - Create processor to sort list
    - Marshall Java Objects back to JSON
- Test in REST client

- Tip: Unmarshall converts JSON into Hashmaps

# Create Restlet to Create Game State

- Define a route to create game state
  - POST game state to **/training-server/rest/game/session**
- Test in game widget (as user)

# Exercise: Active MQ

- Handle Message Asynchronously
  - Queue: Update Game State
    - UpdateGameSession
  - Queue: Update Player High Score
    - UpdatePlayerHighScore
  - Topic: Handle High Score Updated
- Setup JMS Connector
  - Add camel active mq component dependencies
  - Configure Connection Factory using Spring XML
- Create a queue published
  - Create a restlet that forwards the message to a queue using the JMS component
- Create a topic subscriber
  - Log incoming events from training-server to file or console
    - Optional: Display incoming events using Atmosphere

http://camel.apache.org/activemq.html

# Exercise: Web Services

- Player Management SOAP service
  - Login
  - logout
  - getPlayer
- Generate JAX WS classes from WSDL
  - http://localhost:9999/training-server/PlayerManager?wsdl
  - Use Apache CXF wsdl2java maven task
- Define JAX-WS bean in backbase-integration-service.xml
  - Use Spring WS + JaxWsPortProxyFactoryBean
- Create login route
  - Use Bean component
  - Marshall output body from POJO to JSON
- Test with REST client (such as Postman)

http://camel.apache.org/data-format.html

# Exercise: Using Camel Cache

- Optional
    - Cache List of Players for 2 minutes
    - Use camel-cache

http://camel.apache.org/cache.html

BACKBASE

# Thank you!

www.backbase.com
sales-eu@backbase.com

**New York:** +1 646 478 7538
**Amsterdam:** +31 20 465 8888