



Grails Messaging

Architecture

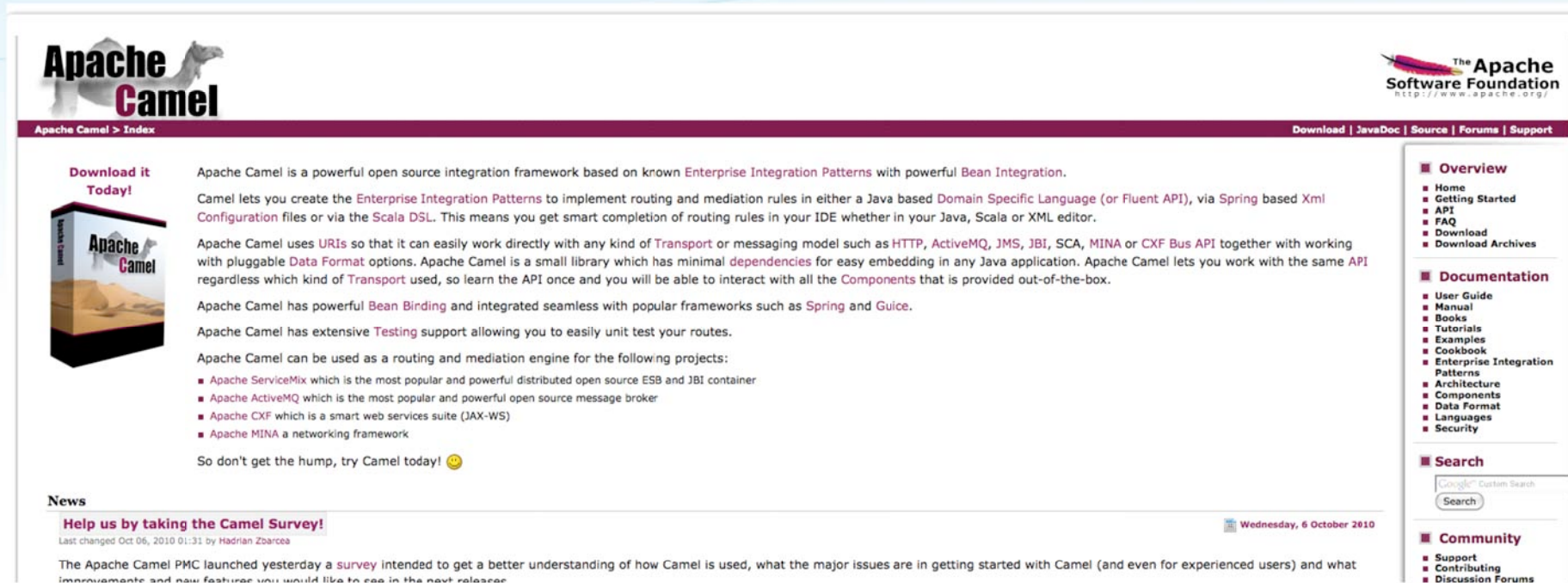
Spring + Grails

Looking for a light-weight container
and flexibility in application building.

Part 2/2: Slides are at:<http://interlated.com.au/presentations/>



Apache Camel and Messaging



The screenshot shows the Apache Camel website homepage. At the top left is the Apache Camel logo featuring a camel. To the right is the Apache Software Foundation logo with the URL <http://www.apache.org/>. Below the logos is a navigation bar with links for [Download](#), [JavaDoc](#), [Source](#), [Forums](#), and [Support](#). The main content area has a heading "Download it Today!" next to a small image of the Apache Camel software box. The text describes Apache Camel as a powerful open source integration framework based on known Enterprise Integration Patterns with powerful Bean Integration. It mentions that Camel lets you create the Enterprise Integration Patterns to implement routing and mediation rules in either a Java based Domain Specific Language (or Fluent API), via Spring based Xml Configuration files or via the Scala DSL. It also states that Apache Camel uses URIs so that it can easily work directly with any kind of Transport or messaging model such as HTTP, ActiveMQ, JMS, JBI, SCA, MINA or CXF Bus API together with working with pluggable Data Format options. A list of projects that use Camel is provided: Apache ServiceMix, Apache ActiveMQ, Apache CXF, and Apache MINA. A "News" section at the bottom left features a link to "Help us by taking the Camel Survey!" with a sub-headline "The Apache Camel PMC launched yesterday a survey intended to get a better understanding of how Camel is used, what the major issues are in getting started with Camel (and even for experienced users) and what improvements and new features you would like to see in the next release." The date "Wednesday, 6 October 2010" is displayed on the right side of the page. A sidebar on the right contains navigation links for Overview, Documentation, Search, and Community.

Apache Camel

Apache Camel > Index

Download | JavaDoc | Source | Forums | Support

Download it Today!

Apache Camel is a powerful open source integration framework based on known [Enterprise Integration Patterns](#) with powerful [Bean Integration](#). Camel lets you create the [Enterprise Integration Patterns](#) to implement routing and mediation rules in either a Java based [Domain Specific Language \(or Fluent API\)](#), via [Spring based Xml Configuration](#) files or via the [Scala DSL](#). This means you get smart completion of routing rules in your IDE whether in your Java, Scala or XML editor.

Apache Camel uses [URIs](#) so that it can easily work directly with any kind of [Transport](#) or messaging model such as [HTTP](#), [ActiveMQ](#), [JMS](#), [JBI](#), [SCA](#), [MINA](#) or [CXF Bus API](#) together with working with pluggable [Data Format](#) options. Apache Camel is a small library which has minimal [dependencies](#) for easy embedding in any Java application. Apache Camel lets you work with the same [API](#) regardless which kind of [Transport](#) used, so learn the [API](#) once and you will be able to interact with all the [Components](#) that is provided out-of-the-box.

Apache Camel has powerful [Bean Binding](#) and integrated seamless with popular frameworks such as [Spring](#) and [Guice](#).

Apache Camel has extensive [Testing](#) support allowing you to easily unit test your routes.

Apache Camel can be used as a routing and mediation engine for the following projects:

- [Apache ServiceMix](#) which is the most popular and powerful distributed open source ESB and JBI container
- [Apache ActiveMQ](#) which is the most popular and powerful open source message broker
- [Apache CXF](#) which is a smart web services suite ([JAX-WS](#))
- [Apache MINA](#) a networking framework

So don't get the hump, try Camel today! 😊

News

[Help us by taking the Camel Survey!](#)

Last changed Oct 06, 2010 01:31 by Hadrian Zbarcea

Wednesday, 6 October 2010

The Apache Camel PMC launched yesterday a [survey](#) intended to get a better understanding of how Camel is used, what the major issues are in getting started with Camel (and even for experienced users) and what improvements and new features you would like to see in the next release.

Overview

- Home
- Getting Started
- API
- FAQ
- Download
- Download Archives

Documentation

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

Search

Google Custom Search

Search

Community

- Support
- Contributing
- Discussion Forums

<http://camel.apache.org>



Middleware

Spring Config (once you get Maven going)

```
<?xml version="1.0" encoding="UTF-8" ?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:camel="http://camel.apache.org/schema/spring"
  xmlns:broker="http://activemq.apache.org/schema/core"
  xmlns:tx="http://www.springframework.org/schema/tx"
  xmlns:util="http://www.springframework.org/schema/util"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd
    http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd
    http://activemq.apache.org/schema/core http://activemq.apache.org/schema/core/activemq-core.xsd
    http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd
    http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util.xsd">

  <!-- START SNIPPET: e2 -->
  <!-- let Spring do its IoC stuff in this package -->
  <context:component-scan base-package="au.com.hearing.server" />

  <!-- PRODUCTION server routes -->
  <bean id="route" class="au.com.hearing.routes.ServerRoutes" />

  <camel:camelContext trace="true" id="camel">
    <camel:package>au.com.hearing.server</camel:package>
    <!-- <camel:jmxAgent id="agent" createConnector="true" /> -->
    <camel:template id="camelTemplate" />
    <camel:routeBuilder ref="route" />
  </camel:camelContext>
```



ActiveMq

```
<import resource="camel-context.xml" />

<broker:broker useJmx="true" persistent="false" brokerName="tcp">
  <broker:transportConnectors>
    <broker:transportConnector name="tcp" uri="${activemq.uri}" />
  </broker:transportConnectors>
</broker:broker>

<bean id="jmsConnectionFactory" class="org.apache.activemq.spring.ActiveMQConnectionFactory">
  <property name="brokerURL" value="${activemq.uri}" />
</bean>

<bean id="pooledConnectionFactory" class="org.apache.activemq.pool.PooledConnectionFactory">
  <property name="maxConnections" value="8" />
  <property name="maximumActive" value="500" />
  <property name="connectionFactory" ref="jmsConnectionFactory" />
</bean>

<bean id="jmsConfig" class="org.apache.camel.component.jms.JmsConfiguration">
  <property name="connectionFactory" ref="pooledConnectionFactory" />
  <property name="transacted" value="false" />
  <property name="concurrentConsumers" value="10" />
</bean>

<bean id="activemqServer" class="org.apache.activemq.camel.component.ActiveMQComponent">
  <property name="configuration" ref="jmsConfig" />
</bean>
```



Camel Routes

```
// All activities for a given date range
from("activemq:queue:checkForNewActivitiesDateSCR")
    .to("log:au.com.hearing.server.debug?showAll=true&multiline=true")
    .to("bean:offlineActivitiesDAO?method=availableActivitiesDate")
    .process(new NoticeSetOutputProcessor());

// changes list - SCR
from("activemq:queue:checkForNewActivitiesDeltaSCR")
    .to("log:au.com.hearing.server.debug?showAll=true&multiline=true")
    .process(new ActivityTypeSCRProcessor())
    .to("bean:offlineActivitiesDAO?method=availableActivitiesDelta")
    .process(new CrbOutputProcessor());

// reconcile a list of notices - SCR
from("activemq:queue:reconcileNoticesSCR")
    .to("log:au.com.hearing.server.debug?showAll=true&multiline=true")
    .process(new ActivityTypeSCRProcessor())
    .to("bean:offlineActivitiesDAO?method=reconcileActivities")
    .process(new CrbOutputProcessor());

from("activemq:queue:healthCheck")
    .to("bean:healthCheck")
    .process(new HealthCheckOutputProcessor());

from("activemq:queue:pingCheck")
    .process(new PingOutputProcessor());

// All bookings for PCAB,
from("activemq:queue:changeRecordClientAppointment")
    .to("log:au.com.hearing.server.debug?showAll=true&multiline=true")
    .doTry()
    .to("bean:createClientDAO?method=create")
    .process(new ChangeRecordClientResponseProcessor())
    .doCatch(AhcisDataIntegrityViolationException.class)
    .process(new ChangeRecordClientExceptionProcessor())
    .end();
```



Grails Messaging

Shared Configuration

Messaging Middleware and
Web Application Share Domain +
Container Configuration





Grails Messaging

Grails Spring

spring/resources.xml

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:camel="http://camel.apache.org/schema/spring"
  xsi:schemaLocation="
    http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-2.5.xsd
    http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd
    http://activemq.apache.org/schema/core http://activemq.apache.org/schema/core/activemq-core-5.3.0.xsd">

  <!-- START SNIPPET: e2 -->
  <!-- let Spring do its IoC stuff in this package -->
  <context:component-scan base-package="au.com.hearing.ahsCaller"/>

  <camel:camelContext trace="true" id="camelClient">
    <camel:package>au.com.hearing.ahsCaller</camel:package>
    <!-- enable JMX connector so we can connect to the server and browse mbeans -->
    <!-- Camel will log at INFO level the service URI to use for connecting with jconsole -->
    <!-- <camel:jmxAgent id="agent" createConnector="true"/> -->

    <!-- for test producers -->
    <camel:template id="camelTemplate"/>
  </camel:camelContext>
</beans>
```



Grails Active MQ

- Copy Camel + ActiveMQ JAR files to lib + spring/resources.groovy

```
beans = {
    switch (GrailsUtil.environment) {
        case "production":

            jmsConnectionFactoryLocal(org.apache.activemq.ActiveMQConnectionFactory) {
                brokerURL = 'vm://localhost'
            }

            jmsConnectionFactory(ActiveMQConnectionFactory) {
                brokerURL = ConfigurationHolder.config.caller.activemqServer
            }

            println "*** broker ${ConfigurationHolder.config.caller.activemqServer}"

            jmsConnectionFactory(org.apache.activemq.pool.PooledConnectionFactory) {
                connectionFactory = ref('jmsConnectionFactory')
            }

            jmsConfig(org.apache.camel.component.jms.JmsConfiguration) {
                connectionFactory = ref('jmsConnectionFactory')
                transacted = true
                concurrentConsumers = 10
            }

            activemqServer(org.apache.activemq.camel.component.ActiveMQComponent) {
                configuration = ref('jmsConfig')
            }

            break
        case "development":
            // vm:(broker:(tcp://localhost:61616)) - ??
            jmsConnectionFactoryLocal(org.apache.activemq.ActiveMQConnectionFactory) {
                brokerURL = 'vm://localhost'
            }

            // jenkins?
    }
```



Grails Messaging

There are many ways...

- Grails Routes – just in java

```
public class Routes extends RouteBuilder {  
  
    @Override  
    public void configure() throws Exception {  
        errorHandler(defaultErrorHandler()  
            .maximumRedeliveries(5).redeliverDelay(60000));  
  
        onException(IOException.class)  
            .maximumRedeliveries(5)  
            .handled(true)  
            .to("bean:userFeedbackService?method=handleErrors");  
  
        Tracer tracer = new Tracer();  
        tracer.setLogLevel(LoggingLevel.ERROR);  
  
        // and we must remember to add the tracer to Camel  
        getContext().addInterceptStrategy(tracer);  
  
        // Reconcile all types on startup, once  
        // TVR  
        from("timer:reconcileTVR?delay=60000&period=86400000")  
            .errorHandler(defaultErrorHandler().maximumRedeliveries(0))  
            .doTry()  
            .to("bean:noticeReconciliationService?method=reconcileTVR")  
            .to("bean:changeRecordBatchService?method=updateDelta")  
            .to("bean:noticeReconciliationService?method=checkPartialBatchTVR")  
            .doCatch(JmsException.class)  
            .process(new DeltaErrorProcessor()).to("bean:userFeedbackService?method=handleDeltaErrors")  
            .end();  
  
        from("seda:checkPartialBatchTVR")  
            .errorHandler(defaultErrorHandler().maximumRedeliveries(0))  
            .doTry()  
            .to("bean:noticeReconciliationService?method=callPartialBatchTVR")  
            .to("bean:changeRecordBatchService?method=updateDelta")  
            .to("bean:noticeReconciliationService?method=checkPartialBatchTVR")  
            .doCatch(JmsException.class)  
            .process(new DeltaErrorProcessor()).to("bean:userFeedbackService?method=handleDeltaErrors")  
            .end();  
  
        // TVN  
        from("timer:reconcileTVN?delay=60000&period=86400000")  
            .errorHandler(defaultErrorHandler().maximumRedeliveries(0))  
            .doTry()  
            .to("bean:noticeReconciliationService?method=reconcileTVN")  
            .to("bean:changeRecordBatchService?method=updateDelta")  
            .to("bean:noticeReconciliationService?method=checkPartialBatchTVN")  
            .doCatch(JmsException.class)  
            .process(new DeltaErrorProcessor()).to("bean:userFeedbackService?method=handleDeltaErrors")  
            .end();  
    }  
}
```





Grails Messaging

Grails Camel Plugin

```
class ContentRoutes {  
    // TODO - array would work  
    // "rss:${ConfigurationHolder.config.emissionsCalculator.content.feeds[0]}?consumer.delay=360000&splitEntries=false"  
  
    def routes = {  
        'rss:http://twitter.com/statuses/user_timeline/102562151.rss?consumer.delay=900000&consumer.initialDelay=60000&split  
        to 'bean:rssReceiverService?method=receive'  
    }  
  
    'direct:testCustomDSL' {  
        to 'bean:rssReceiverService?method=receive'  
    }  
  
    // update topics method - could write a processor to put the vocabulary id in the body.  
  
    'timer:topicTipUpdater?period=3600000&delay=90000&fixedRate=true' {  
        to 'bean:topicService?method=updateTopicsTips'  
    }  
  
    'timer:topicContentUpdater?period=3600000&delay=95000&fixedRate=true' {  
        to 'bean:topicService?method=updateTopicsContent'  
    }  
    }  
}
```





Grails Domain Objects

- Defined in Middleware – mvn package & copy jar file

```
* @author jrobens
*/
@Component
@Entity(name = "Notice")
@XmlRootElement(name = "notice")
@XmlAccessorType(XmlAccessType.FIELD)
public class Notice extends ChangesParentImpl implements Serializable {
    private static Log LOG = LogFactory.getLog(Notice.class);

    private static Set<String> SYSTEM_COLUMNS;

    static {
        SYSTEM_COLUMNS = new HashSet<String>();
        SYSTEM_COLUMNS.add("getId");
        SYSTEM_COLUMNS.add("getDateCreated");
        SYSTEM_COLUMNS.add("getLastUpdated");
        SYSTEM_COLUMNS.add("getArchived");
    }

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    @XmlElement(name = "NOTICE_ID")
    private String noticeId;

    // Only want to save changes to community - might not even want to do that - it should be a static value.
    // Definitely not delete.
    // Not nullable.
    @XmlElement(name = "COMMUNITY")
    @ManyToOne(cascade = {CascadeType.PERSIST, CascadeType.MERGE}, fetch = FetchType.EAGER)
    @PrimaryKeyJoinColumn
    private Community_Val community;
}
```





Grails Messaging

Grails: Please Use Hibernate

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE hibernate-configuration PUBLIC
    "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
    "http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">
<hibernate-configuration>
  <session-factory>
    <mapping package="au.com.hearing.server.domain" />
    <mapping class="au.com.hearing.server.domain.Action" />
    <mapping class="au.com.hearing.server.domain.ChangeRecord" />
    <mapping class="au.com.hearing.server.domain.Client" />
    <mapping class="au.com.hearing.server.domain.Community_Val" />
    <mapping class="au.com.hearing.server.domain.ModifiedField" />
    <mapping class="au.com.hearing.server.domain.Notice" />
    <mapping class="au.com.hearing.server.domain.UpdateScoreboard" />

    <!--
    Not needed on client
    <mapping class="au.com.hearing.server.domain.Reconcile" />
    <mapping class="au.com.hearing.server.domain.Repository" />
    <mapping class="au.com.hearing.server.domain.Subscriber" />
    -->
    <mapping class="au.com.hearing.server.domain.VersionVal" />
    <mapping class="au.com.hearing.server.domain.MarketingActivity" />
  </session-factory>
</hibernate-configuration>
```





Grails Messaging

Grails Service to Handle Message

```
/**
 * Provide a service interface to ChangeRecordBatch related routes.
 */
class ChangeRecordBatchService {
    boolean transactional = true
    def camelTemplate

    def updateDelta(ChangeRecordBatch crb) {
        // check that we have a batch
        if (crb == null) {
            LOG.error("updateDelta: Null change record batch.")
            return null
        }

        AuditLog auditLog = auditLogService.log(crb.id.toString(), ChangeRecordBatch.getCanonicalName(), "ChangeRecordBatch")
        camelTemplate.sendBody("seda:auditLog", auditLog)

        crb.changeRecords.sort {it.crID.toLong()}.eachWithIndex {cr, i ->
```





InOut Pattern – sending messages

```
AuditLog auditLog = auditLogService.log("", Notice.getCanonicalName(), "NoticeService.updateBulk", null, Boolean.TRUE)
camelTemplate.sendBody("seda:auditLog", auditLog)

List<Notice> response = new ArrayList<Notice>()
Notice.withTransaction { status ->
    String requestTimeout = ConfigurationHolder.config.caller.requestTimeout ? ConfigurationHolder.config.caller.requestT
    response = camelTemplate.requestBody("activemqServer:queue:checkForNewActivities?requestTimeout=${requestTimeout}", n
}
return response
```



Grails Messaging

Publish Subscribe

- Cost us

```
from("activemqServer:topic:notice.updates")  
    .to("bean:noticeService?method=updates")
```





Large Batches

```
ChangeRecordBatch.withTransaction { status ->
    deltaRouter(cr)

    if (i % commitBatchAfter == 0) {
        log.info ">> running session clear after ${i}"
        sessionFactory.getCurrentSession().clear()
    }
}
```



Grails Messaging

Grails + Messaging

- Shared Configuration
- Find a way that works
- Patterns and Synchronicity

