Progress | FUSE

Apache Camel

Migrating to Version 2.5

Version 2.5
October 2010

Migrating to Version 2.5

Version 2.5

Publication date 18 Oct 2010

Legal Notices

Third Party Acknowledgments

Apache Camel 2.5 incorporates ANTLR, Another Tool for Language Recognition v2.7.7 from Terrence Parr. Such technology is subject to the following terms and conditions: SOFTWARE RIGHTS ANTLR 1989-2006 Developed by Terence Parr Partially supported by University of San Francisco & jGuru.com. We reserve no legal rights to the ANTLR--it is fully in the public domain. An individual or company may do whatever they wish with source code distributed with ANTLR or the code generated by ANTLR, including the incorporation of ANTLR, or its output, into commercial software. We encourage users to develop software with ANTLR. However, we do ask that credit is given to us for developing ANTLR. By "credit", we mean that if you use ANTLR or incorporate any source code into one of your programs (commercial product, research project, or otherwise) that you acknowledge this fact somewhere in the documentation, research report, etc... If you like ANTLR and have developed a nice tool with the output, please mention that you developed it using ANTLR. In addition, we ask that the headers remain intact in our source code.

As long as these guidelines are kept, we expect to continue enhancing this system and expect to make other tools available as they are completed. The primary ANTLR guy: Terence Parr parrt@cs.usfca.edu parrt@antlr.org.

Apache Camel 2.5 incorporates ANTLR, Another Tool for Language Recognition v3.0.1 from Terence Parr. Such technology is subject to the following terms and conditions: [The "BSD licence"] Copyright © 2003-2006 Terence Parr All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE AUTHOR ``AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Apache Camel 2.5 incorporates Apache ORO v2.0.8 from the Apache Foundation. Such technology is subject to the following terms and conditions: Apache Software License Version 1.1 Copyright © 2000 The Apache Software Foundation. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the Apache Software Foundation (http://www.apache.org/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 4. The names "Apache" and "Apache Software Foundation" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact apache@apache.org. 5. Products derived from this software may not be called "Apache", nor may "Apache" appear in their name, without prior written permission of the Apache Software Foundation. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. This software consists of voluntary contributions made by many individuals on behalf of the Apache Software Foundation. For more information on the Apache Software Foundation, please see http://www.apache.org/. Portions of this software are based upon public domain software originally written at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign.

Apache Camel 2.5 GA incorporates ASM v1.5.3 from INRIA, France Telecom. Such technology is subject to the following terms and conditions: ASM: a very small and fast Java byte code manipulation framework Copyright © 2000, 2002, 2003 INRIA, France Telecom All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of the copyright holders nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND

CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Apache Camel 2.5 GA incorporates JDOM v1.1 from Jason Hunter& Brett McLaughlin. Such technology is subject to the following terms and conditions: jdom License - Copyright © 2000-2004 Jason Hunter & Brett McLaughlin. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions, and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions, and the disclaimer that follows these conditions in the documentation and/or other materials provided with the distribution. 3. The name "JDOM" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact {request_AT_jdom_DOT_org}. 4. Products derived from this software may not be called "JDOM", nor may "JDOM" appear in their name, without prior written permission from the JDOM Project Management {request_AT_jdom_DOT_org}. In addition, we request (but do not require) that you include in the end-user documentation provided with the redistribution and/or in the software itself an acknowledgement equivalent to the following: "This product includes software developed by the JDOM Project (http://www.jdom.org/)." Alternatively, the acknowledgment may be graphical using the logos available at http://www.jdom.org/images/logos. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE JDOM AUTHORS OR THE PROJECT CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Apache Camel 2.5 incorporates OSCore – opensymphony v2.2.4, OSWorkflow v2.7.0, PropertySet – opensymphony v1.3. Such technologies are subject to the following terms and conditions: The OpenSymphony Software License, Version 1.1 (this license is derived and fully compatible with the Apache Software License - see http://www.apache.org/LICENSE.txt) Copyright © 2001-2004 The OpenSymphony Group. All rights reserved. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 1. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 2. The end-user documentation included with the redistribution, if any, must include the following acknowledgment: "This product includes software developed by the OpenSymphony Group (http://www.opensymphony.com/)." Alternately, this acknowledgment may appear in the software itself, if and wherever such third-party acknowledgments normally appear. 3. The names "OpenSymphony" and "The OpenSymphony Group" must not be used to endorse or promote products derived from this software without prior written permission. For written permission, please contact license@opensymphony.com. 4. Products derived from this software may not be called "OpenSymphony" or "OSCache", nor may "OpenSymphony" or "OSCache" appear in their name, without prior written permission of the OpenSymphony Group. THIS SOFTWARE IS PROVIDED "AS IS" AND ANY EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE APACHE SOFTWARE FOUNDATION OR ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY

OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Apache Camel 2.5 incorporates Saxon XSLT and XQuery Processor – saxon v9.1.0.1. The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is all Saxon modules labeled with a notice referring to this license. The Initial Developer of the Original Code is Michael Kay, except where otherwise specified in an individual module. Portions created by other named contributors are copyright as identified in the relevant module. All Rights Reserved. Contributor(s) are listed in the documentation: see notices/contributors.

Apache Camel 2.5 incorporates Rhino: JavaScript for Java v1.7R1. The contents of this file are subject to the Mozilla Public License Version 1.1 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is Rhino code, released May 6, 1999. The Initial Developer of the Original Code is Netscape Communications Corporation. Portions created by the Initial Developer are Copyright © 1997-1999 the Initial Developer. All Rights Reserved. Contributor(s).

Table of Contents

List of Tables

Chapter 1. Apache Camel Issues

Spring Framework

Spring version

Since Apache Camel 2.5, Apache Camel uses Spring 3.0.4 as the default Spring version. Spring 2.5 is also still supported, but Spring 2.0.x is now deprecated and, in future releases, will *not* be supported.

New schema location

If you explicitly specify the location of the Spring schema in your Spring configuration files, you must change the schema location to point at the 3.0 Spring schema. The Spring 3.0 is located at the following Web page:

```
http://www.springframework.org/schema/beans/spring-beans-
3.0.xsd
```

For example, assuming your schema locations are specified in the root `beans` element, you could specify the new Spring schema location as follows:

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans ht
tp://www.springframework.org/schema/beans/spring-beans-
3.0.xsd">
```

Order of dependency injection in Spring 3.0

It appears that the order in which beans are dependency injected has changed in Spring 3.0. This could potentially affect your existing Apache Camel applications when you upgrade. To gain more control over the order of dependency injection, you could add the `depends-on` attribute to some of your bean definitions.

Spring 3.0 new features

For a summary of the new features in Spring 3.0, see New Features and Enhancements in Spring 3.0[1].

[1] http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/new-in-3.html

Product Dependencies

Java development kit

Since Apache Camel 2.5, the Apache Camel Web Console requires JDK1.6 or newer.

To build or run ordinary applications developed with Apache Camel, JDK 1.5.0_11 or newer is sufficient.

Maven version

Since Apache Camel 2.5, you require Maven 2.2.1 or later to build the source distribution of Apache Camel.

Apache CXF version

Since Apache Camel 2.5, the CXF component is updated to use Apache CXF version 2.2.11.

Namespace Changes

Namespace for the Spring DSL schema

The XML namespace for the Spring DSL schema has changed between Apache Camel 1.x and Apache Camel 2.2, as follows:

Old XML schema namespace:

```
http://activemq.apache.org/camel/schema/spring
```

New XML schema namespace:

```
http://camel.apache.org/schema/spring
```

Moreover, when specifying the `xsi:schemaLocation` attribute, you need to specify the location of the *new* XML schema.

Example

You need to update all of your old Spring XML configuration files to use the new schema and the new schema location. For example, the namespace settings (which are typically defined in a Spring `bean` element) should be changed as follows:

Old namespace definitions in `bean` element:

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:camel="http://act
ivemq.apache.org/camel/schema/spring"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans ht
tp://www.springframework.org/schema/beans/spring-beans-2.5.xsd

         http://activemq.apache.org/camel/schema/spring ht
tp://activemq.apache.org/camel/schema/spring/camel-spring.xsd">
```

New namespace definitions in `bean` element:

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:camel="http://camel.apache.org/schema/spring"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans ht
tp://www.springframework.org/schema/beans/spring-beans-2.5.xsd

         http://camel.apache.org/schema/spring ht
tp://camel.apache.org/schema/spring/camel-spring.xsd">
```

JAR Dependencies

JAR dependencies

The following JAR dependencies have changed:

- From Apache Camel 2.4 onwards, the OSGi integration code and the Spring integration code have been decoupled, so that there is no longer any need for the following artifacts:
  - `camel-osgi`
  - `camel-spring-osgi`

  In fact, these artifacts are no longer provided with Apache Camel 2.4. If you want to use the integration with Spring, you only need to install the `camel-spring` artifact.
- From Apache Camel 2.5 onwards, the following test artifacts have been deprecated and will be removed in a future release:
  - `camel-core-tests`
  - `camel-spring-tests`

  In future, you should use only the `camel-test` artifact, if you want to use the Apache Camel test kit.

DSL Changes

DSL changes

Table 1.1 on page 16 provides an overview of the DSL commands that have been renamed in Apache Camel 2.0.

Table 1.1. Renamed DSL Commands

Old Java DSL Name	Old Spring DSL Name	New DSL Name
`splitter`	`splitter`	`split`
`resequencer`	`resequencer`	`resequence`
`aggregator`	`aggregator`	`aggregate`
`delayer`	`delayer`	`delay`
`throttler`	`throttler`	`throttle`
`expression`	`expression`	`language`
`tryBlock`	`try`	`doTry`
`handle`	`catch`	`doCatch`
`finallyBlock`	`finally`	`doFinally`
`intercept`	`intercept`	`interceptFrom`
`thread`	`thread`	`threads`
`bean` (in `SpringBuilder` class)		`lookup`
`throwFault`		*Removed*

errorHandler changes

The following changes were made to the `errorHandler` DSL command in Apache Camel version 2.1:

- In the Java DSL the, `errorHandler` DSL command can now be configured only on a route or a Camel context. In the case of routes, you must set it directly after the `from` DSL command.

- In Spring DSL, the `errorHandlerRef` attribute is now available only on the `camelContext` and `route` elements.

---

adviceWith changes

Since Apache Camel version 2.3, the `adviceWith()` DSL command now takes `CamelContext` as its first argument.

API Changes

API changes

The following changes have been made to the Java API:

ProducerTemplate

Since Apache Camel 2.0, the `org.apache.camel.ProducerTemplate` class has been refactored so that `sendBody` methods now return void for *InOnly* messaging. Use one of the `requestBody` methods for *InOut* messaging. See "Producer and Consumer Templates" in *Programmer's Guide* for more details.

Exchange

Since Apache Camel 2.0, all specializations of `org.apache.camel.Exchange` are now removed. You should now always use the `org.apache.camel.impl.DefaultExchange` type.

After analyzing the how exchanges are used, it was realized that the specialized exchanges were not really necessary and by removing them we can avoid a lot of unnecessary copying and improve throughput.

One implication of this change is that `Producer` and `Consumer` types no longer use generic type declarations. For example, instead of referring to `Producer<DefaultExchange>` you can just refer to a plain `Producer`.

Exchange `getFault()` and `setFault()` methods

Since Apache Camel 2.0, the `getFault()` and `setFault()` methods are now removed from `Exchange`. Faults represent application specific errors and are recognized by some protocols. Consequently, it makes more sense to store a fault as the *Out* message in an exchange (accessed using `getOut()` and `setOut()`). The `org.apache.camel.Message` interface now exposes the boolean `isFault()` and `setFault()` methods that are used to identify *Out* messages that represent faults.

Because faults represent persistent errors (as opposed to exceptions, which represent transient errors), Camel does not try (as in previous versions) to recover from them (for example, the error handler does not trigger) unless handling faults as exceptions is explicitly enabled.

AggregationStrategy

Since Apache Camel 2.0, the following changes have been made to the implementation of `org.apache.camel.processor.aggregate.AggregationStrategy`:

- The `aggregate()` method is now invoked on the very first exchange. For this first invocation, the `oldExchange` parameter is `null`.
- The payload is now always stored in the *In* message when you do custom aggregation using this strategy interface.

Aggregator

Since Apache Camel 2.3, the aggregator has been re-implemented and some options have been replaced. In particular, the algorithms for determining batch completeness have changed significantly, so that the `batchSize`, `outBatchSize`, `batchTimeout`, and `batchConsumer` options are no longer supported. To understand the new mechanisms for completeness testing, it is recommended that you read "Aggregator" in *Implementing Enterprise Integration Patterns*.

CamelContext

Since Apache Camel version 2.1, the following changes have been made to the `CamelContext` class:

- The `shouldStartContext()` method is replaced by the `autoStartup()` method.
- The `getLifecycleStrategy()` method has been renamed `getLifecycleStrategies()` and now returns a `java.util.List`.

Since Apache Camel version 2.5, the following changes have been made to the `CamelContext` class:

- The `stopRoute()` method is now integrated with the graceful shutdown strategy (see "Controlling Start-Up and Shutdown of Routes" in *Implementing Enterprise Integration Patterns*). You can revert to the old behavior by specifying an explicit timeout.

ManagementNamingStrategy

Since Apache Camel version 2.1, the `org.apache.camel.spi.ManagementNamingStrategy` has had

methods renamed and method signatures changed in order to accomodate the JMX features in this release.

PollingConsumerPollStrategy
: Since Apache Camel version 2.3, the `begin()` method from `org.apache.camel.spi.PollingConsumerPollStrategy` returns a boolean value, where `true` indicates that polling can now start, while `false` indicates that polling should be skipped.

RoutePolicy
: Since Apache Camel version 2.3, the `org.apache.camel.spi.RoutePolicy` interface has the new method, `onInit()`.

Message
: Since Apache Camel version 2.3, the `org.apache.camel.Message` interface has the new method, `removeHeaders()`.

DefaultComponent and DefaultEndpoint
: Since Apache Camel version 2.3, the `getExecutorService()` and `setExecutorService()` methods have been removed from the `org.apache.camel.impl.DefaultComponent` and `org.apache.camel.impl.DefaultEndpoint` classes. To create a thread pool, use the `ExecutorServiceStrategy` object that is returned by the `CamelContext.getExecutorServiceStrategy()` method.

  For full details of the new threading model, see "Threading Model" in *Implementing Enterprise Integration Patterns*.

GenericFile
: Since Apache Camel version 2.3, the `org.apache.camel.component.file.GenericFile` class is no longer serializable (does not inherit from `java.io.Serializable`).

RouteDefinition
: Since Apache Camel version 2.3, the `adviceWith()` method from `org.apache.camel.model.RouteDefinition` takes a `CamelContext` instance as its first parameter.

toAsync
: Since Apache Camel version 2.4, the `toAsync()` DSL command has been removed. Asynchronous dispatch is now implemented directly (where appropriate) in specific Apache Camel components and DSL commands.

Policy
: Since Apache Camel version 2.4, the `org.apache.camel.spi.Policy` interface has the new method, `beforeWrap()`. For quick migration of your `Policy` classes, simply add an empty method implementation.

onException
: Since Apache Camel version 2.4, the `retryUntil` option on `onException` has been renamed to `retryWhile`, because this reflects the meaning of the option more accurately (it continues to retry while its argument is true).

Routing Slip
: Since Apache Camel version 2.4, you can use either a string or an expression to specify the name of the routing slip header. The Spring DSL has changed, such that the `headerName` attribute is now replaced by the `headerName` child element. For example, to specify the name of the routing slip header to be `myHeader`, use an XML fragment like the following:

```
<route>
  <from uri="direct:a"/>
  <routingSlip ignoreInvalidEndpoints="true">
    <headerName>myHeader</headerName>
  </routingSlip>
</route>
```

ProducerTemplate
: Since Apache Camel version 2.4, all `sendBody` and `requestBody` methods from the `ProducerTemplate` class throw a `CamelExecutionException`, which wraps the original exception.

RouteBuilder
: Since Apache Camel version 2.4, the `simple` and `xpath` expression builder methods are built into the `RouteBuilder` class. It is, therefore, no longer necessary to use static imports to access these languages. In

your existing code, the Java compiler might complain, if you use static imports for `simple` and `xpath`. To fix this, just remove the static imports.

Removed classes

The following classes have been removed from the API:

```
org.apache.camel.processor.CompositeProcessor
org.apache.camel.impl.ProducerTemplateProcessor
org.apache.camel.impl.NoPolicy
org.apache.camel.spi.Provider
org.apache.camel.spring.handler.LazyLoadingBeanDefinitionParser

org.apache.camel.spring.handler.ScriptDefinitionParser
org.apache.camel.spring.remoting.SendBeforeInterceptor
org.apache.camel.spring.spi.SpringConverters
```

Moved classes

The following classes have moved to a different Java package:

- `PollingConsumerPollStrategy` has moved from `org.apache.camel` to `org.apache.camel.spi`.
- `PatternBasedPackageScanFilter` has moved from `org.apache.camel.impl.scan` to `org.apache.camel.spring`.

The following classes have been renamed:

- `org.apache.camel.management.event.ExchangeFailureEvent` has been renamed to `ExchangeFailedEvent`.

Component Updates

File and FTP components

Since Apache Camel version 2.0, the following changes have been made to the File and the FTP components:

- Only a directory name can be specified directly in the endpoint, *not a filename*. In other words, you must specify a File endpoint with the syntax, `file:directoryName[?options]` and an FTP component with the syntax, `ftp://[username@]hostname[:port]/directoryname[?options]`. It is still possible to select individual files, however, by appending the `fileName` option, which enables you to specify files using the file expression language[2].
- File producer endpoints and FTP producer endpoints now overwrite existing files by default (previously, the default behavior was to append to files).You can use the new `fileExist` option to specify what happens when a producer attempts to write a file that already exists. Valid values for the `fileExist` option are: `Override`, `Append`, `Fail`, or `Ignore`.

Important

Both the File component and the FTP component have been extensively rewritten in Apache Camel 2.0. In particular, many of the component options have been renamed or modified. It is recommended that you check the configuration of your File and FTP endpoints against the latest component documentation—see "File2" in *Component Reference* and "FTP2" in *Component Reference*.

JMS component

The JMS `correlationId` is determined differently to before. Now, JMS always uses the `messageId` as the `correlationId`, if configured to do so by setting `useMessageIDAsCorrelationID` to `true`. If the setting is `false`, the `JMSCorrelationID` header value is used, if present, and the `messageId` value is used as a fallback, if the header is not present.

Since Apache Camel version 2.5, the following changes have been made to the JMS component:

- The JMS 1.0.2 API is no longer supported.

[2] http://camel.apache.org/file-language.html

- Durable topic subscribers now must provide a `clientId` value, otherwise Apache Camel fails fast on start-up.

List component	The List component has been renamed to Browse.
Bean component	The Bean component now enforces stricter matching of bean method parameters. Previously, if a parameter could not be converted to the appropriate type, it was passed as `null`. Now, all parameters must be convertible to the relevant types.
Direct component	The `allowMultipleConsumers` option has been removed. A Direct endpoint can now have only *one* consumer.
HTTP component	Since Apache Camel version 2.2, HTTP proxy configuration is set using `CamelContext` properties instead of Java system properties. Since Apache Camel version 2.3, the HTTP authentication options have been renamed, to avoid clashing with the `username` and `password` URI parameters. Authentication is now set using the `authUsername`, `authPassword`, and `authDomain` options. In addition, you must specify the authentication method using the `authMethod` option, which can take the values `Basic`, `Digest`, or `NTLM`. For the full list of HTTP authentication options, see "HTTP" in *Component Reference*.
SEDA and VM components	Since Apache Camel 2.0, these components support request/reply semantics. Endpoints of SEDA or VM type will wait for a reply, if a reply is expected. Since Apache Camel 2.3, the size of the SEDA queue (which holds incoming exchanges) is unbounded, whereas previously it had a maximum size of 1000.
MINA component	Since Apache Camel version 2.3, the header key, `MinaConsumer.HEADER_CLOSE_SESSION_WHEN_COMPLETE`, is replaced by `MinaConstants.MINA_CLOSE_SESSION_WHEN_COMPLETE`.
Jetty component	Since Apache Camel version 2.3, the Jetty component has been upgraded from Jetty 6.1.22 to Jetty 7.0.1. The Jetty API has changed significantly

between these two versions. Because Jetty is now hosted at Eclipse[3], all Java packages have been renamed. If you use the Jetty API directly, you could use the Jetty 6 to Jetty 7 migration tool[4] from Eclipse to simplify the migration of your source code.

CXF component

Since Apache Camel version 2.3, the CXF component's `PAYLOAD` mode has been improved to delegate all SOAP message parsing to CXF.

Since Apache Camel version 2.5, the CXF component is based on Apache CXF 2.2.11.

FTP component

Since Apache Camel version 2.4, the following changes have been made to the FTP component:

- The `ftps` default port has been changed from 2222 to 21.
- The `ftps` protocol now uses a secure data channel while transferring files, whereas previously only secure login was enabled. You can now use the `disableSecureDataChannelDefaults`, `execProt`, and `execPbsz` options to customize the behavior of the security channel. See "FTP2" in *Component Reference* for details.
- The FTP base directory can now be specified using an absolute path. In the endpoint URI, insert two leading forward slashes, `//`, to specify an absolute path, as in `ftp:admin@someserver//absolutePath/foo/bar?password=secret`.
- The FTP component now uses a 10 second default connect timeout (for all protocols) and a 30 second data timeout (for the `ftp` and `ftps` protocols only).

Since Apache Camel version 2.5, the following changes have been made to the FTP component:

[3] http://www.eclipse.org
[4] http://wiki.eclipse.org/Jetty/Howto/Upgrade_from_Jetty_6_to_Jetty_7

- FTP consumer endpoints now traverse the file structure in a different way. It is recommended that you test any applications with FTP consumer endpoints to ensure that they are not affected by this change.

Netty component

Since Apache Camel 2.5, the `timeout` option on the Netty component has been removed, because it did not work properly.

Quartz component

Since Apache Camel 2.5: if you are using the Quartz component with jobs persisted in a database, you should note that Apache Camel now resolves job names based on the endpoint URI *without* parameters. This makes it possible to change cron parameters on the same job (that is, to reschedule the job).

Miscellaneous Changes

Error handling

Apache Camel 2.2 uses the new `DefaultErrorHandler`, instead of the `DeadLetterChannel` (versions 1.x), as the default error handler. This new `DefaultErrorHandler` implementation does *not* catch and handle thrown exceptions, which means any exception thrown will be propagated back to the caller.

Stream caching

Stream caching is a feature that enables you to re-read a message body consisting of a stream type. The interaction between Apache ServiceMix and Apache Camel has now been modified in such a way that Apache Camel does *not* need to use stream caching. It is therefore disabled by default.

Note

If necessary, you can re-enable caching by adding the `streamCaching()` command to a route in the Java DSL or by setting the `route` element's `streamCaching` attribute to `true` in the Spring XML DSL.

Annotations

The following changes have been made to the Java annotations:

- The `name` attribute in `@EndpointInject` has been changed to `ref`, in order to be consistent with the other annotations (where the `@EndpointInject` annotation references an Endpoint that gets looked up in the Registry).
- Since Apache Camel 2.3, the spelling of the `parallelProcessing` attribute in `@RecipientList` has been fixed.

Case-insensitivity of header look-up

Since Apache Camel 2.0, header look-up in `org.apache.camel.Message` is case-insensitive. This means that if you look up a header using differently cased keys (such as `Foo` and `foo`), you will obtain a reference to the *same*

header entry. This makes header look-up less error-prone when used with protocols such as HTTP, where header names are inherently case insensitive.

Round robin load balancing

The round robin load balancing behavior has changed in Apache Camel 2.2. In version 1.x, the round robin load balancer would fail over, if an endpoint failed to process the message. In version 2.2, the round robin load balancer does not fail over immediately, but tries to redeliver the message to the same endpoint, according to the error handling configuration.

Note

Apache Camel 2.3 will introduce an option that allows you to specify which behavior you want the endpoint to exhibit.

Splitter

Since Apache Camel 2.3, if the exchange that enters the splitter has the *InOut* message-exchange pattern (that is, a reply is expected), the splitter returns a copy of the original input message as the reply message in the *Out* message slot. To revert to the old (pre 2.3) behavior, configure the splitter to use the `UseLatestAggregationStrategy` aggregation strategy.

xpath language and XML conversions

Since Apache Camel 2.3, the `NodeList` to `String` converter has been improved to include XML tags and attributes. For example, the conversion can now produce strings like `<foo id="123">bar<year>2015</year></foo>`.

This change will affect code that relied on the old string conversion. For example, if you previously extracted strings from XML using `@XPath` annotations as in the following method signature:

```
// Java
// Camel version 2.2 or earlier:
public void credit(
       @XPath("/transaction/transfer/receiver") String name,

        @XPath("/transaction/transfer/amount") String amount
        )
```

You must now explicitly specify the `text()` node, in order to avoid including the element tags in the string, as follows:

```
// Java
// Camel version 2.3 or later
public void credit(
```

```
      @XPath("/transaction/transfer/receiver/text()") String
name,
       @XPath("/transaction/transfer/amount/text()") String
amount
       )
```

Converter character set

The character set used by Apache Camel converters can be set using the `org.apache.camel.default.charset` Java system property. The default is `UTF-8`.

CamelContext

Since Apache Camel 2.5, the following changes affect `CamelContext`:

- Apache Camel fails to start up, if more than one `CamelContext` instance with the same ID is registered in JMX.
- If you do not specify an ID value for a `camelContext` element, Apache Camel now assigns an automatically generated ID from the sequence, `camel-1`, `camel-2`, and so on. In previous Apache Camel versions, the ID value defaulted to the string, `camelContext`.

UuidGenerator

Since Apache Camel 2.5, Apache Camel supports the use of third party UUID generators. The default UUID generator is based on the same implementation used in Apache ActiveMQ, `ActiveMQUuidGenerator`. Some of the APIs required by this implementation (for example, the Google app engine) might not be accessible in all deployment contexts, in which case you should switch to another of the UUID generator implementations. The following implementations are provided:

- `org.apache.camel.impl.JavaUuidGenerator`.
- `org.apache.camel.impl.SimpleUuidGenerator`.
- `org.apache.camel.impl.ActiveMQUuidGenerator`.

You can set the UUID generator in either of the following ways:

- In the Java DSL, call the `CamelContext.setUuidGenerator()` method as follows:

```
// Java
getContext().setUuidGenerator(new MyCustomUuidGenerator());
```

- In the Spring DSL, simply instantiate a bean of the requisite type. For example, to enable the `JavaUuidGenerator`:

```
<bean id="javaUuidGenerator"
      class="org.apache.camel.impl.JavaUuidGenerator" />
```