Camel User Guide

Apache ServiceMix
Version 7.0.0-SNAPSHOT

1. Introduction

Apache Camel is a powerful open source integration framework based on known Enterprise Integration Patterns with powerful Bean Integration.

1.1. Camel in ServiceMix

In ServiceMix, Apache Camel is like our swiss army knife for creating integration solutions. It allows using XML or a Java/Scala-based DSL to express your routes, comes with over 70 optional components, has powerful and versatile Java bean integration, error handling, ... and tons of other features.

Apache Camel is installed by default if you first start the container. We also have out-of-the-box hot-deployment support for both Spring and Blueprint to make it easy to deploy your own Camel routes, as well as optionally installable features for all the available Camel components.

1.2. Goal of this guide

The goal of this guide is to look into the details for using Camel inside ServiceMix:

- deployment options
- installing additional components

1.3. Examples

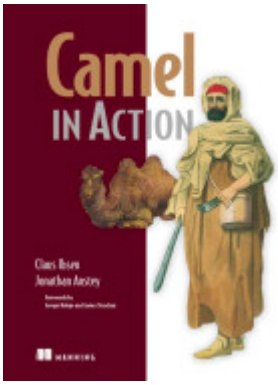
The Apache ServiceMix distributions also contain a set of Camel examples. You can find these examples in the `examples/camel` directory.

1.4. More information about Camel

More information about Camel itself, can be found on http://camel.apache.org.

There's also a great book available about Camel

- Ibsen, Claus, and Anstey, Jonathan. (December 2010). *Camel in Action*. Greenwich, CT: Manning. ISBN: 9781935182368.



2. Deployment options

There are a few different ways to deploy Camel routes on ServiceMix 7.0.0-SNAPSHOT:

- deploy routes in a plain Blueprint XML file
- deploy routes in a plain Spring XML file
- deploy a bundle containing a Blueprint XML file
- deploy a bundle containing a Spring XML file

Camel routes can also be deployed as part of a JBI SA, allowing you use Camel for routing between JBI endpoints - this option will be discussed later when we are talking about using JBI inside ServiceMix 4.

Benefits and drawbacks

Plain XML or OSGi bundles

Choose a plain XML file:

- if you want to get routes deployed as quickly as possible
  all you need for developing routes is a simple text editor, no compilation, building, ... required at all
- if you prefer the XML syntax over the Java of Scala DSL

Choose an OSGi bundle:

- if you want to package helper classes together with your route definitions
- if you prefer developing routes in the Java or Scala DSL
  you can package the RouteBuilder implementations inside the bundle

Blueprint or Spring

Choose Blueprint:

- if you want the best possible integration with the OSGi Framework and Service Registy
  the Blueprint specification has been developed specifically for the OSGi Framework by the OSGi Alliance

Choose Spring:

- if you already invested in Spring for creating and running Camel routes

2.1. Deploy as a plain Spring XML file

ServiceMix 7.0.0-SNAPSHOT supports the deployment of plain Spring XML files, automatically creating and starting the Spring ApplicationContext from the XML file.

In order to leverage this feature to create and start Camel routes, drop a file with this syntax in the `$SERVICEMIX_HOME/deploy` folder:

```
<?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: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://camel.apache.org/schema/spring
            http://camel.apache.org/schema/spring/camel-spring-2.10.3.xsd">

  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <!-- add Camel routes, interceptors,... here -->
  </camelContext>

</beans>
```

An example

Just create a new XML file in the deploy folder with the code below to start a route to copy files from one directory to another.

```
<?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: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://camel.apache.org/schema/spring
            http://camel.apache.org/schema/spring/camel-spring-2.10.3.xsd">

  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
      <from uri="file:input"/>
      <log message="Copying ${file:name} to the output directory"/>
      <to uri="file:output"/>
    </route>
  </camelContext>

</beans>
```

2.2. Deploy as a plain Blueprint XML file

ServiceMix 7.0.0-SNAPSHOT supports the deployment of plain Blueprint XML files, automatically creating and starting the Blueprint container from the XML file.

In order to leverage this feature to create and start Camel routes, drop a file with this syntax in the `$SERVICEMIX_HOME/deploy` folder:

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint
    xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
      http://www.osgi.org/xmlns/blueprint/v1.0.0
      http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
        <!-- add Camel routes, interceptors,... here -->
    </camelContext>

</blueprint>
```

An example

Just create a new XML file in the deploy folder with the code below to start a route to copy files from one directory to another.

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint
    xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
      http://www.osgi.org/xmlns/blueprint/v1.0.0
      http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
      <route>
        <from uri="file:input"/>
        <log message="Copying ${file:name} to the output directory"/>
        <to uri="file:output"/>
      </route>
    </camelContext>

</blueprint>
```

2.3. Deploy as an OSGi bundle with Spring

Using an OSGi bundle to deploy your Camel routes allows you to use the Java or Scala DSL for defining your routes.

In this case, you're using Spring to start your Camel routes, so you include your Spring XML file (e.g. `camel-context.xml`) in the `META-INF/spring` folder inside your bundle.

```
+ <bundle classes, incl. your RouteBuilder>
\- META-INF
   |- MANIFEST.MF
   \- spring
      \- camel-context.xml
```

After the bundle has been activated, the Spring DM extender will find, create and start your Spring ApplicationContexts.

Example: Referring to Java or Scala RouteBuilder classes

If your RouteBuilder classes have been defined in the `org.apache.servicemix.manual.camel` package, the file would look like this:

```
<?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: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://camel.apache.org/schema/spring
            http://camel.apache.org/schema/spring/camel-spring-${camel-version}.xsd">

  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <package>org.apache.servicemix.manual.camel</package>
  </camelContext>

</beans>
```

Example in the distribution

Another example for using this deployment option can be found in the `camel-osgi` example that is shipped with Apache ServiceMix.

2.4. Deploy as an OSGi bundle

Using an OSGi bundle to deploy your Camel routes allows you to use the Java or Scala DSL for defining your routes.

In this case, we will use a Blueprint XML file to start your Camel routs. To do so, the Blueprint XML files have to be included in the bundle inside the `OSGI-INF/blueprint` directory.

```
+ <bundle classes, incl. your RouteBuilder>
|- META-INF
|  |- MANIFEST.MF
\- OSGI-INF
   \- blueprint
      \- camel-context.xml
```

As soon as the bundle becomes Active, the Blueprint extender will create the Blueprint container starting your Routes.

Example: Referring to Java or Scala RouteBuilder classes

If your RouteBuilder classes have been defined in the `org.apache.servicemix.manual.camel` package, the file would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<blueprint
    xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
      http://www.osgi.org/xmlns/blueprint/v1.0.0
      http://www.osgi.org/xmlns/blueprint/v1.0.0/blueprint.xsd">

    <camelContext xmlns="http://camel.apache.org/schema/blueprint">
      <package>org.apache.servicemix.manual.camel</package>
    </camelContext>

</blueprint>
```

Example in the distribution

Another example for using this deployment option can be found in the `camel-blueprint` example that is shipped with Apache ServiceMix.

3. Installing components

Camel comes with over 80 components, so you can imagine that we don't install all of them by default. This section shows you how to find available components and how to install them at runtime.

List available components

Camel components are available as installable features. You can look at the full list of available features using the `features:list` command, using `grep` to limit things down to features related to camel:

```
karaf@root> features:list | grep camel
[installed  ] [2.10.3    ] camel                                 repo-0
[installed  ] [2.10.3    ] camel-core                            repo-0
[installed  ] [2.10.3    ] camel-spring                          repo-0
[installed  ] [2.10.3    ] camel-blueprint                       repo-0
[uninstalled] [2.10.3    ] camel-test                            repo-0
[uninstalled] [2.10.3    ] camel-cxf                             repo-0
[uninstalled] [2.10.3    ] camel-cache                           repo-0
[uninstalled] [2.10.3    ] camel-castor                          repo-0
...
```

The items marked with **`installed`** in the first column have already been installed and are available for use in your Camel routes.

Install and uninstalling components

You can use `features:install` to install any component on the list.

An example: to install the `camel-cache` component

```
karaf@root> features:install camel-cache
```

Similarly, you can also uninstall components that you're no longer using with `features:uninstall`

```
karaf@root> features:uninstall camel-cache
{pygementize}
```

4. Troubleshooting

In this section, you'll find solutions for some frequently asked questions when using Camel on ServicMix.

No component with id 'xyz' could be found

This usually means that your route is trying to use a component that hasn't been installed yet.

Solution:

1. install the additional component
2. restart the bundle using the `bundle:restart <bundle id>` command - you can find the bundle id for your route in the output of the `bundle:list` command

Refer to Installing additional components for more information about installing additional components.