WildFly Camel

This is the umbrella project for Apache Camel integration with the WildFly Application Server.

It is the home of the Camel subsystem, integration test suite, installer, documentation, etc.

Features

Camel Context Definitions

Camel Contexts can be defined as part of the WildFly Camel subsystem definition.

Camel Context Deployments

Camel Contexts can be deployed as single XML file or part of another deployment supported by WildFly.

Camel Feature Provisioning

WildFly Camel provides feature provisioning similar to Eclipse or Karaf. It uses Gravia to provide this functionality.

Integration with JAX-WS

WebService support is provided through the camel-cxf component which integrates with the WildFly WebServices subsystem.

Integration with JMS

Messaging support is provided through the camel-jms component which integrates with the WildFly Messaging subsystem.

Integration with JNDI

Naming support is provided through the WildFlyCamelContext which integrates with the WildFly Naming subsystem.

Integration with JMX

Management support is provided through the camel-jmx component which integrates with the WildFly JMX subsystem.

Arquillian Test Support

WildFly Camel uses Arquillian to connect to an already running WildFly server or start one up when needed.

User Guide

This chapter takes you through the first steps of getting WildFly Camel and provides the initial pointers to get up and running.

Download the Distribution

[TODO]

Installing the Camel Subsystem

[TODO]

Camel Context Definitions

Camel Contexts can be configured in standalone-camel.xml as part of the subsystem definition like this

```
<subsystem xmlns="urn:jboss:domain:camel:1.0">
  <camelContext id="system-context-1">
    <![CDATA[
    <route>
      <from uri="direct:start"/>
      <transform>
        <simple>Hello #{body}</simple>
      </transform>
    </route>
    ]]>
  </camelContext>
</subsystem>
```

On WildFly startup you should see something like this

```
10:01:29,213 INFO  [org.wildfly.camel] (MSC service thread 1-7) JBAS020001: Register camel context: sy
10:01:29,214 INFO  [org.wildfly.camel] (MSC service thread 1-1) JBAS020002: Bound camel naming obje
```

Camel Context Deployments

There are two ways to deploy a Camel Context to WildFly

1. As a single XMl file with a predefined **-camel-context.xml** file suffix
2. As part of another WildFly supported deployment as **META-INF/jboss-camel-context.xml** file

When deployed as XML file, you should see

```
10:20:01,621 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-3) JBAS015876: Starting dep
...
10:20:01,893 INFO  [org.apache.camel.spring.SpringCamelContext] (MSC service thread 1-1) Apache Car
...
10:20:01,945 INFO  [org.apache.camel.spring.SpringCamelContext] (MSC service thread 1-1) Route: rout
10:20:01,949 INFO  [org.apache.camel.spring.SpringCamelContext] (MSC service thread 1-1) Apache Car
10:20:01,955 INFO  [org.wildfly.camel] (MSC service thread 1-1) JBAS020001: Register camel context: sp
...
10:20:01,963 INFO  [org.jboss.as.server] (management-handler-thread - 7) JBAS018559: Deployed "simp
```

When deployed as part of another deployment, you should something similar

```
10:24:02,649 INFO  [org.jboss.as.server.deployment] (MSC service thread 1-6) JBAS015876: Starting dep
...
10:24:02,882 INFO  [org.apache.camel.spring.SpringCamelContext] (MSC service thread 1-1) Apache Car
...
10:24:02,935 INFO  [org.apache.camel.spring.SpringCamelContext] (MSC service thread 1-1) Route: rout
10:24:02,940 INFO  [org.apache.camel.spring.SpringCamelContext] (MSC service thread 1-1) Apache Car
10:24:02,945 INFO  [org.wildfly.camel] (MSC service thread 1-1) JBAS020001: Register camel context: sp
...
10:24:02,952 INFO  [org.jboss.as.server] (management-handler-thread - 11) JBAS018559: Deployed "cam
```

Camel Feature Provisioning

WildFly Camel provides feature provisioning similar to Karaf features. A feature is defined as set of abstract Resources with associated Capabilities/Requirements. All known features are stored in a Repository. At runtime the Provisioner gets a set of Resource candidates from the Repository and uses the Resolver to find a consistent wiring solution for the current state of the Environment. After this no-impact analysis, the Provisioner installs the required set of Resources to the Environment if a consistent wiring solution can be found by the Resolver.

The initial set of supported features is part of the WildFly Camel repository content definition. Resources that are already part of the WildFly environment are defined as part of the environment content

A good starting point to work with WildFly Camel feature provisioning is ProvisionerSupport and references to it.

The concepts of Resource, Capability, Requirement, Resolver, Repository, Provisioner are all provided by the Gravia project, which is a rewrite of the same functionality that used to be available in WildFly as part of the JBoss OSGi integration.

WildFly Camel feature provisioning has no dependency on OSGi.

```
ProvisionerSupport provisionerSupport = new ProvisionerSupport(provisioner);
provisionerSupport.installCapabilities(IdentityNamespace.IDENTITY_NAMESPACE, "camel.cxf.feature");
...
```

Integration with JAX-WS

WebService support is provided through the camel-cxf component which integrates with the WildFly WebServices subsystem that also uses Apache CXF.

```
// Create the CamelContext
CamelContext camelctx = contextFactory.createWildflyCamelContext(getClass().getClassLoader());
camelctx.addRoutes(new RouteBuilder() {
   @Override
   public void configure() throws Exception {
      from("direct:start").
      to("cxf://" + getEndpointAddress("/simple") + "?serviceClass=" + Endpoint.class.getName());
   }
});
camelctx.start();

ProducerTemplate producer = camelctx.createProducerTemplate();
String result = producer.requestBody("direct:start", "Kermit", String.class);
Assert.assertEquals("[Hello Kermit]", result);
```

Integration with JMS

Messaging support is provided through the camel-jms component which integrates with the WildFly Messaging (HornetQ) subsystem.

```
// Create the CamelContext
CamelContext camelctx = contextFactory.createWildflyCamelContext(getClass().getClassLoader());
camelctx.addRoutes(new RouteBuilder() {
   @Override
   public void configure() throws Exception {
      from("jms:queue:" + QUEUE_NAME + "?connectionFactory=ConnectionFactory").
      transform(body().prepend("Hello ")).to("direct:end");
   }
});
camelctx.start();

// Send a message to the queue
ConnectionFactory cfactory = (ConnectionFactory) initialctx.lookup("java:/ConnectionFactory");
Connection connection = cfactory.createConnection();
sendMessage(connection, QUEUE_JNDI_NAME, "Kermit");

String result = consumeRouteMessage(camelctx);
Assert.assertEquals("Hello Kermit", result);
```

Integration with JNDI

The WildFlyCamelContext provides integration with the WildFly Naming subsystem.

```
WildflyCamelContext camelctx = contextFactory.createWildflyCamelContext(getClass().getClassLoader());

// Bind a bean to JNDI
Context context = camelctx.getNamingContext();
context.bind("helloBean", new HelloBean());
camelctx.addRoutes(new RouteBuilder() {
   @Override
   public void configure() throws Exception {
      from("direct:start").beanRef("helloBean");
   }
});
camelctx.start();

ProducerTemplate producer = camelctx.createProducerTemplate();
String result = producer.requestBody("direct:start", "Kermit", String.class);
Assert.assertEquals("Hello Kermit", result);

context.unbind("helloBean");
```

Integration with JMX

Management support is provided through the camel-jmx component which integrates with the WildFly JMX subsystem.

```
CamelContext camelctx = contextFactory.createWildflyCamelContext(getClass().getClassLoader());
camelctx.addRoutes(new RouteBuilder() {
   @Override
   public void configure() throws Exception {
      String host = InetAddress.getLocalHost().getHostName();
      from("jmx:platform?format=raw&objectDomain=org.apache.camel&key.context=" + host + "/syster
      "&monitorType=counter&observedAttribute=ExchangesTotal&granularityPeriod=500").
      to("direct:end");
   }
});
camelctx.start();

ConsumerTemplate consumer = camelctx.createConsumerTemplate();
MonitorNotification notifcation = consumer.receiveBody("direct:end", MonitorNotification.class);
Assert.assertEquals("ExchangesTotal", notifcation.getObservedAttribute());
```

Arquillian Test Support

The WildFly Camel test suite uses the WildFly Arquillian managed container. This can connect to an already running WildFly instance or alternatively start up a standalone server instance when needed.

A number of test enrichers have been implemented that allow you have these WildFly Camel specific types injected into your Arquillian test cases.

```
@ArquillianResource
CamelContextFactory contextFactory;

@ArquillianResource
CamelContextRegistry contextRegistry;
```

Developer Guide

Source

https://github.com/tdiesler/wildfly-camel

Issues

https://github.com/tdiesler/wildfly-camel/issues

Jenkins

http://174.129.32.31:8080/job/tdi-wildfly-camel

Downloads

[TODO]

https://repository.jboss.org/nexus/content/groups/public-jboss/org/wildfly/camel

Forums, Lists, IRC

[TODO]

#fabric8 channel on irc.freenode.net

Table of Contents