CXF Tomcat Example

CXF Tomcat Example

Available as of Camel 2.5

This example is located in the `examples/camel-example-cxf-tomcat` directory of the Camel distribution.
There is a `README.txt` file with instructions how to run it.

If you use maven then you can easily package the example from the command line:

```
mvn package
```

About

This example demonstrates how you can use CXF to expose a web service in Camel using code first approach.

Implementation

The web service we want to expose is defined as an interface which has 2 operations:

Error rendering macro 'code': Invalid value specified for parameter 'java.lang.NullPointerException'

```
public interface IncidentService {

    /**
     * Operation to report an incident
     */
    OutputReportIncident reportIncident(InputReportIncident input);

    /**
     * Operation to get the status of an incident
     */
    OutputStatusIncident statusIncident(InputStatusIncident input);
}
```

In this example we are not using any JAX-WS annotations. You can use those annotations to fine control the web service wsdl contract.

In the Camel route we expose this web service very easily using the Camel CXF component.
All we have to do is to define an endpoint uri in the format

```
cxf:/incident?serviceClass=org.apache.camel.example.cxf.incident.IncidentService
```

This means Camel will expose the web service using the relative address `/incident` and the serviceClass parameter links to the interface which defines the code first approach.

In this example we want to be flexible, so if we add a 3rd operation to the web service we want it to be easily to add a route to handle this operation. Therefore we use the Recipient List EIP pattern to route to the route which handles the given operation. Notice how we use a Direct endpoint to link the routes.

Error rendering macro 'code': Invalid value specified for parameter 'java.lang.NullPointerException'

```
public class CamelRoute extends RouteBuilder {

    // CXF webservice using code first approach
    private String uri = "cxf:/incident?serviceClass=" + IncidentService.class.getName();

    @Override
    public void configure() throws Exception {
        from(uri)
            .to("log:input")
            // send the request to the route to handle the operation
            // the name of the operation is in that header
            .recipientList(simple("direct:${header.operationName}"));

        // report incident
        from("direct:reportIncident")
            .process(new Processor() {
                public void process(Exchange exchange) throws Exception {
                    // get the id of the input
                    String id = exchange.getIn().getBody(InputReportIncident.class).getIncidentId();

                    // set reply including the id
                    OutputReportIncident output = new OutputReportIncident();
                    output.setCode("OK;" + id);
                    exchange.getOut().setBody(output);
                }
            })
            .to("log:output");

        // status incident
        from("direct:statusIncident")
            .process(new Processor() {
                public void process(Exchange exchange) throws Exception {
                    // set reply
                    OutputStatusIncident output = new OutputStatusIncident();
                    output.setStatus("IN PROGRESS");
                    exchange.getOut().setBody(output);
                }
            })
            .to("log:output");
    }
}
```

Spring XML

In the Spring XML file we have to import some CXF mandatory imports. Notice we use the `cxf-servlet` to leverage HTTP Servlet with CXF.

Error rendering macro 'code': Invalid value specified for parameter 'java.lang.NullPointerException'

```
<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"
       xmlns:cxf="http://camel.apache.org/schema/cxf"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
       http://camel.apache.org/schema/spring http://camel.apache.org/schema/spring/camel-spring.xsd
       http://camel.apache.org/schema/cxf http://camel.apache.org/schema/cxf/camel-cxf.xsd">
  <!-- We still need it for loading the CXFServlet -->
  <import resource="classpath:META-INF/cxf/cxf.xml"/>

  <bean id="myRoutes" class="org.apache.camel.example.cxf.CamelRoute"/>

  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <routeBuilder ref="myRoutes"/>
  </camelContext>

</beans>
```

web.xml

In the `web.xml` file we have just to setup Spring and CXF the usual way.

Error rendering macro 'code': Invalid value specified for parameter 'java.lang.NullPointerException'

```
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-app_2_4.
xsd">

        <display-name>My Web Application</display-name>

        <!-- location of spring xml files -->
        <context-param>
                <param-name>contextConfigLocation</param-name>
        <param-value>classpath:camel-config.xml</param-value>
        </context-param>

        <!-- the listener that kick-starts Spring -->
        <listener>
                <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>

        <!-- CXF servlet -->
        <servlet>
                <servlet-name>CXFServlet</servlet-name>
                <servlet-class>org.apache.cxf.transport.servlet.CXFServlet</servlet-class>
                <load-on-startup>1</load-on-startup>
                <!-- If you want to leverage the Servlet3's async feature in Tomcat,
                 please enable this feature
                <async-supported>true</async-supported>
                -->
        </servlet>

        <!-- all our webservices are mapped under this URI pattern -->
        <servlet-mapping>
                <servlet-name>CXFServlet</servlet-name>
                <url-pattern>/webservices/*</url-pattern>
        </servlet-mapping>

</web-app>
```

Running the example

This example runs in Apache Tomcat, so you will have to package the .war file and copy it to the `webapp` folder of Tomcat, which is the hot deploy folder.

Note: You have to use the version number of Camel you use. In this documentation we are using 2.5.0.

You can then use SoapUI or another web service client and send a request to the `http://localhost:8080/camel-example-cxf-tomcat-2.5.0/webservices/incident` url.

The wsdl is located at: `http://localhost:8080/camel-example-cxf-tomcat-2.5.0/webservices/incident?wsdl`.
And CXF outputs which web services it has from this url `http://localhost:8080/camel-example-cxf-tomcat-2.5.0/webservices`.

See Also

- Examples
- CXF
- HTTP
- Servlet