Contents

1 CXF Labs

1.1 Warmup Labs

1.1.1 Pre-requisites

Action	Tool	Commands

CLEAN start of SMX	Linux	`cd %SMX_HOME%`
		`bin/stop`
		`bin/start clean`

Watch SMX logs	Linux	`less %SMX_HOME%/data/logs/servicemix.log`
		`"F" > follow`
		`CTRL+C > abort follow`
		`q > quit`

Start SMX console	Linux	`bin/client`
Verify that SMX really has been started cleanly, i.e. only bundles up to id 232 have been installed	SMX	`list`

1.1.2 Lab "Install feature"

1.1.2.1 Goal

The goal is to understand:

- Which features are available for installation?
- How to find out what is included in a feature.

1.1.2.2 Exercise

Action	Tool	Commands	Question
List feature repositories	SMX	`feature:repo-list`	Is there a repository that contains SMX examples?
Show contents of the examples features repo	SMX	`shell:cat mvn:org.apache.servicemix/apache-servicemix/6.1.0/xml/examples`	How many bundles & features does feature "examples-camel-cxf-rest" contain?

Find feature repo file on disk	Linux	`less %SMX_HOME%/system`
		`find . -name \*.xml`
			- Is there a feature repo file for SMX examples?
			- Find how many bundles the "examples-camel-cxf-rest" feature should contain according to that file!
			- Does it give the expected result?

1.1.3 Lab “Modify an example”

1.1.3.1 Goal

The goal is to understand:

- How the prebuilt examples can be deployed to SMX.
- How a code change in an example can be deployed.

1.1.3.2 Exercise

We demonstrate this with the “camel-cxf-rest” example.

Action	**Tool**	**Commands**	**Question**
Go to project	Linux	`cd %SMX_HOME%/examples/camel/camel-cxf-rest`	How many modules does the project contain?
Review README.txt	Linux	`less README.txt`	How to install the prebuilt example?
Install the prebuilt example	SMX	`feature:install examples-camel-cxf-rest`	
List installed bundles	SMX	`bundle:list`	How many bundles did get installed by the feature?

Modify project on disk & build	Linux	`cd camel-cxf-rest-route`
		`vi pom.xml`
		
		change
		- from
		`<name>Apache ServiceMix :: Examples :: Camel CXF REST :: Route</name>`
		- to
		`<name>SMXSMX :: Route</name>`
		
		`mvn clean install`	

Refresh bundle	SMX	`bundle:update <...>`	
Verify that bundle was renamed	SMX	`bundle:list`	Did it get renamed?

1.2 Lab WS Code First (from SMX example ”cxf-jaxws-blueprint”)

1.2.1 Goal

The goal is to understand:

- How is a web service annotated?
- What needs to be configured in blueprint.xml for CXF?

1.2.2 Exercise

1.2.2.1 Pre-requisites

see 1.1.1

1.2.2.2 Install prebuilt feature

Action	Tool	Commands	Question
Go to project	Linux	`cd %SMX_HOME%/examples/cxf/cxf-jaxws-blueprint`	How many modules does the project contain?
Review README.txt	Linux	`less README.txt`	How to install the prebuilt example?
Install the prebuilt example	SMX	`feature:install examples-cxf-jaxws-blueprint`	
List installed bundles	SMX	`bundle:list`	How many bundles did get installed by the feature?

1.2.2.3 Run prebuilt client

Action	Tool	Commands	Question
Run client and verify that the bundle is working	Linux	`mvn compile exec:java`	Was a SOAP response received?

1.2.2.4 Run manual curl commands

Action	Tool	Commands	Question
Verify in Firefox that REST service is available	Firefox	`localhost:8181/cxf`	On which port is the SOAP service available?
Run POST SOAP	Linux	`curl -X POST -T ./src/main/resources/org/apache/servicemix/samples/cxf_osgi/request.xml -H "Content-Type: text/xml" http://localhost:8181/cxf/HelloWorld`	Did it get a response?

1.2.2.5 Analyze code within Eclipse

Action	Tool	Commands
Import project into Eclipse	Eclipse	`File > Import > Existing Maven Projects > %SMX_HOME%/examples/cxf/cxf-jaxws-blueprint/pom.xml`

Class/File	Question	Answer
Interface HelloWorld/ class HelloWorldImpl	1) Do they have any JAX-WS annotations? 2) Is the annotation required in both interface and implementation class?	1) Yes 2) Yes. Even though it seems to work in this particular case, generally, annotations are not inherited in Java.
blueprint.xml > <jaxws:endpoint>	1) What does < jaxws:endpoint> do? 2) What does <jaxws:outInterceptors> do? 3) What is the purpose of an interceptor in CXF? 4) What does EnableCORSInterceptor do? 5) There can be multiple interceptors in a chain. How is the execution order determined? In which phase should EnableCORSInterceptor get executed?	1) The jaxws:endpoint element configures a JAX-WS service providers. Its children are used to configure interceptors and other advanced features. 2) It defines an outgoing interceptor, i.e. an interceptor that is executed before sending out the response. 3) Interceptors are the fundamental processing unit inside CXF. When a service is invoked, an InterceptorChain is created and invoked. Each interceptor gets a chance to do what they want with the

		message. This can include reading it, transforming it, processing headers, validating the message, etc. 4) It adds headers to the outgoing SOAP response. 5) Interceptors are run in a certain phase. In which phase it should be run is defined in the constructor. For instance EnableCORSInterceptor is run in the PRE_PROTOCOL phase.
blueprint.xml > <cxf:bus>	1) What is the purpose of <cxf:bus>? 2) What is the purpose of <cxf:features>? 3) What is the purpose of <cxf:logging/>? Remove <cxf:logging/>, build and deploy it. Can you see any changes in the logs happening when sending in a POST request?	1) <cxf:bus> configuration applies to all CXF endpoints. Thus you can define global configuration. 2) <cxf:features> adds features to the <cxf:bus>. Again, this is global. If one wants to add a feature to a certain endpoint than one would have to add <jaxws:features> to the specific endpoint. 3) <cxf:logging/> causes it to log the complete message when receiving requests & sending responses.

1.4 Lab WS Contract First (from archetype ”cxf-wsdl-first-example”)

1.4.1 Goal

The goal is to understand:

- How are the Java classes generated from the WSDL?
- What do the generated classes look like?
- What needs to be configured in beans.xml for CXF?

1.4.2 Exercise

1.4.2.1 Pre-requisites

see 1.1.1

1.4.2.2 Create project from Maven archetype

Action	Tool	Commands	Question	Answer
Go to Eclipse workspace	Linux	`cd %ECLIPSE_WORKSPACE%`		
Create project	Linux	`mvn archetype:generate -DarchetypeGroupId=org.apache.servicemix.tooling -DarchetypeArtifactId=servicemix-cxf-wsdl-first-osgi-bundle -DgroupId=se.test -DartifactId=cxf-wsdl-first-example -Dversion=1.0-SNAPSHOT`	1) What source & properties files do exist?	1) There are only 3 files: • PersonImpl.java • beans.xml • person.wsdl
Build project	Linux	`mvn clean install`	1) Is there still the same number of source files?	1) No, the wsdl2java generator has generated a number of files from the WSDL.

1.4.2.3 Analyze the wsdl2java generated classes within Eclipse

Action	Tool	Commands
Import project into Eclipse	Eclipse	`File > Import > Existing Maven Projects > ...`

Class/File	Question	Answer
Interface Person	1) What purpose does @WebService have?	1) This is a service endpoint interface (SEI) which declares the methods that a client can invoke on the service.
Class PersonService	1) Why does it extend the javax.xml.ws.Service class? 2) What is the purpose of @WebServiceClient?	1) A Service class provides the client view of a web service. 2) In order to allow an implementation to identify the Web service that a generated service class corresponds to, the latter is required to be annotated with javax.xml.ws.WebServiceClient annotation. The annotation contains all the information necessary to locate a WSDL document.
Class UnknownPersonFault	1) What is the purpose of @WebFault?	1) The WebFault annotation is used when mapping WSDL faults to Java exceptions. The annotated class contains a member which is a JAXB type to be able to convert the fault into XML.
Classes ...types.GetPerson ...types.GetPersonResponse ...types.UnknownPersonFault	1) What is the purpose of @XmlRootElement?	1) These are JAXB annotations to convert Java objects into XML.

1.4.2.4 Deploy and troubleshoot the bundle

Action	Tool	Commands	Question	Answer
Build project	Linux	`mvn clean install`		
Deploy project	SMX	`install -s mvn:se.test/cxf-wsdl-first-example/1.0-SNAPSHOT`	1) What does the "-s" option do? 2) Did the bundle get deployed and started?	1) It tries to start the bundle after deployment. 2) No. 3) bundle:diag <...> 4) The problem is "Unsatisfied

			3) How do you diagnose the problem? 4) What is the problem? 5) How can you see which imports and exports the bundle has?	Requirements”, i.e. the bundle wants to import packages that are not exported by any other bundle. 5) bundle:headers <...>
Fix the problem	Linux	The “Unsatisfied Requirements” problem is not really a problem. These imports are no longer required in the later CXF versions (see https://issues.apache.org/jira/browse/SM-2464). Therefore solve this by removing the 2 wrong imports from the maven-bundle-plugin in the pom.xml. mvn clean install		
Update the bundle & start it	SMX	bundle:update <...> bundle:start <...>	1) Did the bundle start?	1) Yes.
Test the bundle	Firefox	http://localhost:8181/cxf	1) Is the web service alive?	1) Yes.

Send a request to the web service	Linux	cat > request.xml
		<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:typ="http://servicemix.apache.org/samples/wsdl-first/types">
		   <soapenv:Header/>
		   <soapenv:Body>
		      <typ:GetPerson>
		         <typ:personId>222</typ:personId>
		      </typ:GetPerson>
		   </soapenv:Body>
		</soapenv:Envelope>
		
		CTRL+D
		
		curl -X POST -T request.xml -H "Content-Type: text/xml" http://localhost:8181/cxf/PersonService
			1) Did you receive any valid SOAP response?	1) Yes.

1.5 Lab REST (from SMX example ”cxf-jaxrs-blueprint”)

1.5.1 Goal

The goal is to understand:

- How are JAXB & CXF annotations being used for REST services?
- What needs to be configured in blueprint.xml for CXF?

1.5.2 Exercise

1.5.2.1 Pre-requisites

see 1.1.1

1.5.2.2 Install prebuilt feature

Action	Tool	Commands	Question
Go to project	Linux	`cd %SMX_HOME%/examples/cxf/cxf-jaxrs-blueprint`	How many modules does the project contain?
Review README.txt	Linux	`less README.txt`	How to install the prebuilt example?
Install the prebuilt example	SMX	`feature:install examples-cxf-jaxrs-blueprint`	
List installed bundles	SMX	`bundle:list`	How many bundles did get installed by the feature?

1.5.2.3 Run prebuilt client

Action	Tool	Commands	Question
Run client and verify that the bundle is working	Linux	`mvn compile exec:java`	Did a POST, GET & PUT get executed?

1.5.2.4 Run manual curl commands

Action	Tool	Commands	Question
Verify in Firefox that REST service is available	Firefox	`localhost:8181/cxf`	On which port is the REST service available?
Run GET	Linux	`curl -v http://localhost:8181/cxf/crm/customerservice/customers/123`	Did it find a person?

Run POST	Linux	`cd src/main/resources/org/apache/servicemix/examples/cxf/jaxrs/client`
		`cat add_customer.xml`
		`curl -X POST -T src/main/resources/org/apache/servicemix/examples/cxf/jaxrs/client/add_customer.xml -H "Content-Type: text/xml" http://localhost:8181/cxf/crm/customerservice/customers`
			What HTTP response code did the POST return?

Run 2nd GET		see above	Did it find a person this time?
Run PUT	Linux	`curl -v -X PUT -T update_customer.xml -H "Content-Type: text/xml" http://localhost:8181/cxf/crm/customerservice/customers`	
Run 2nd GET		see above	Has the person changed?
Run DELETE	Linux	`curl -v -X DELETE http://localhost:8181/cxf/crm/customerservice/customers/123`	What was the HTTP response code?
Run 3rd GET		see above	Did it find a person this time?

1.5.2.5 Analyze code within Eclipse

Action	Tool	Commands
Import project into Eclipse	Eclipse	`File > Import > Existing Maven Projects > %SMX_HOME%/examples/cxf/cxf-jaxrs-blueprint/pom.xml`

Class/File	Question	Answer
Class Customer	1) Does it have any JAXB annotations? 2) What is the purpose of the class?	2) Yes 3) Annotation @XmlRootElement maps a class or an enum type to an XML element.
Class CustomerService	1) Does it have any JAXRS annotations? 2) What is the purpose of this class? 3) What is the purpose of annotations @Api, @ApiOperations, …?	1) Yes 2) It is being used by the CXF REST server to process incoming REST requests.

		3) These are annotations for a tool called Swagger which serves interactive documentation of REST APIs.
blueprint.xml > <bean>	1) Which classes does it instantiate?	1) CustomerService
blueprint.xml > <jaxrs:server>	1) What does <jaxrs:server> do? 2) What does <jaxrs:serviceBeans> do? 3) How does it resolve which service bean to call for a certain REST call? 4) What does <jaxrs:provider> do?	1) <jaxrs:server> creates a JAXRS server which listens on a certain address. 2) It defines service beans that handle REST requests. There can be more than one such service bean. 3) The service beans have @Path(...) annotations which resolve this. 4) Providers are a way of extending and customizing the JAX-RS runtime. They are like plugins that alter the behavior of the runtime, for instance for mapping between representations and their associated Java types.

1.6 Lab CAMEL CXF REST (from SMX example “camel-cxf-rest”)

1.6.1 Goal

The goal is to understand:

- How are JAXB & CXF annotations being used for REST services?
- How does camelContext use CXF REST endpoints?
- What needs to be configured in blueprint.xml for Camel & CXF?

1.6.2 Exercise

1.6.2.1 Pre-requisites

see 1.1.1

1.6.2.2 Install prebuilt feature

Action	Tool	Commands	Question
Go to project	Linux	`cd %SMX_HOME%/examples/camel/camel-cxf-rest`	How many modules does the project contain?
Review README.txt	Linux	`less README.txt`	How to install the prebuilt example?
Install the prebuilt example	SMX	`feature:install examples-camel-cxf-rest`	
List installed bundles	SMX	`bundle:list`	How many bundles did get installed by the feature?

1.6.2.3 Run prebuilt client

Action	Tool	Commands	Question

Run client and verify that the bundle is working	Linux	`cd camel-cxf-rest-client`
		`mvn compile exec:java`
			Did a POST, GET & DELETE get executed?

1.6.2.4 Run manual curl commands

Action	Tool	Commands	Question
Verify in Firefox that REST service is available	Firefox	`localhost:8181/cxf`	On which port is the REST service available?

Run GET	Linux	curl -v -X GET 127.0.0.1:8989/rest/personservice/person/get/1	Did it find a person?
Run POST	Linux	create XML file: cat > smx_post.xml <?xml version="1.0" encoding="UTF-8" standalone="yes"?> <person> <age>21</age> <id>1</id> <name>John Smith</name> </person> CTRL+D curl -v -X POST 127.0.0.1:8989/rest/personservice/person/post -H "Content-Type: application/xml" -T smx_post.xml	What HTTP response code did the POST return?
Run 2nd GET		see above	Did it find a person this time?
Run DELETE	Linux	curl -v -X DELETE 127.0.0.1:8989/rest/personservice/person/delete/1	What was the HTTP response code?
Run 2nd DELETE		see above	What was the HTTP response code this time?
Run 3rd GET		see above	Did it find a person this time?

1.6.2.5 Analyze code within Eclipse

Action	Tool	Commands
Import project into Eclipse	Eclipse	File > Import > Existing Maven Projects > %SMX_HOME%/examples/camel/camel-cxf-rest/pom.xml

1.6.2.5.1 Analyze “camel-cxf-rest-service”

Class/File	Question	Answer
Class Person	1) Does it have any JAXB annotations? 2) What is the purpose of the class?	1) Yes 2) Annotation @XmlRootElement maps a class or an enum type to an XML element.
Class ServiceHandler	What is the purpose of the class?	Holds and manages a list of Person.
blueprint.xml	1) Which class does it instantiate? 2) How many routes does it define in the camel context?	1) ServiceHandler 2) 3 3) direct-vm

	3) Which kind of endpoint do the routes consume? 4) What is the difference between “direct” & “direct-vm” endpoints? 5) <bean method=”getPerson” ...> > How does it know on which class it should call the method? 6) What is “`${header.id}`”? 7) What does “`<transform><simple>${header.id}</simple></transform>`” do?	4) “direct-vm” can be used to connect existing routes in the same camel context, as well from other camel contexts in the same JVM. 5) attribute “ref” 6) This is Apache Commons OGNL (Object-Graph Navigation Language) notation. It accesses header with the key “id” in the IN message of the message exchange. 7) It replaces the body of the IN message with the contents from header “id”.

1.6.2.5.2 Analyze “camel-cxf-rest-route”

Class/File	**Question**	**Answer**
MANIFEST.MF	1) Is there a MANIFEST.MF file? 2) Which packages does the bundle import? 3) Which packages does the bundle export?	1) Not in “src” but in “target”. It got generated by the “maven-bundle-plugin” (see pom.xml). 2) & 3) See bundle:headers <...>
pom.xml > maven-bundle-plugin > Import	1) What does the “*” mean in `<Import-Package>*;resolution:=optional</Import-Package>`?	1) Make sure that you include the wildcard, *, in the Import-Package element of the Maven bundle plug-in configuration. The wildcard directs the plug-in to scan your Java source code and automatically generates a list of package dependencies.
MANIFEST.MF	1) What does “`[2.0,3)`” mean?	1) It requires version 2.0.3 or higher.
blueprint.xml > <cxf:rsServer>	1) What does attribute ‘loggingFeatureEnabled=”true”’ mean? 2) Run a manual GET (as above) operation. > Does the SMX logfile show any LoggingInInterceptor & LoggingOutInterceptor messages? 3) Change this to “false”. > Build and deploy this to SMX. > Retry GET operation. > Does the SMX logfile show any LoggingInInterceptor & LoggingOutInterceptor messages?	1) It enables the CXF logging interceptors. 2) Yes 3) No
class PersonService	1) Does it have any JAXRS annotations? 2) What is the purpose of this class?	1) Yes

		2) It is being used by the CXF REST server to process incoming REST requests.
blueprint.xml > camelContext	1) Why does it send it to `<recipientList>` instead of `<to>`?	1) `<to>` is not capable to route to a dynamic URI. `<recipientList>` can. See http://camel.apache.org/how-to-use-a-dynamic-uri-in-to.html.
blueprint.xml > property-placeholder	1) What does <cxf:rsServer> create? 2) On which URL does the CXF server listen? 3) What is "`cxfrs`" in "`cxfrs:bean:rsServer`"? What is the purpose of component "`cxfrs`"? 4) What is "`bean`" in "`cxfrs:bean:rsServer`"? What is the purpose of component "`bean`"? 5) What is "`rsServer`" in "`cxfrs:bean:rsServer`"?	1) It instantiates a special bean which is the server endpoint to a cxf-rs consumer. 2) localhost:8989/rest 3) The cxfrs: component provides integration of camel with Apache CXF for connecting to JAX-RS 1.1 and 2.0 services hosted in CXF. 4) The bean: component binds beans to Camel message exchanges. 5) "rsServer" is the ID of the actual bean being called.