Onlineversion der Seminararbeit:
Apache CXF (2011)

Weiter Informationen, sowie das Kolloquium zur Seminararbeit, findet sich unter:
http://www.herrmann-online.info/index.php/wissenschaftliche-arbeiten/apache-cxf

Die Arbeit ist Teil einer geplanten Print-Veröffentlichung und versteht sich in der hier dargestellten Form als Arbeitsstand der noch angepasst wird.

Wichtiger Hinweis:
Alle Inhalte wurden sorgfältig geprüft und nach bestem Wissen erstellt. Aber für die hier dargebotenen Informationen wird kein Anspruch auf Vollständigkeit, Aktualität, Qualität und Richtigkeit erhoben. Es kann keine Verantwortung für Schäden übernommen werden, die durch das Vertrauen auf die Inhalte oder deren Gebrauch entstehen. Dies gilt speziell, aber nicht ausschließlich, für ältere Fach- und Studienarbeiten.

http://www.herrmann-online.info
Martin Herrmann, B. Sc.

6 Apache CXF

Martin Herrmann

Inhaltsverzeichnis

6.1 Einleitung

Im Zeitalter des Web 2.0 werden eine Vielzahl von Anwendungen verteilt über das Internet realisiert. Für die Interaktion zwischen Anwendungsprogrammen werden vermehrt Web Services eingesetzt, die eine Zusammenarbeit von Anwendungen auf verschiedenen Plattformen erleichtern sollen. Apache CXF ist ein Framework, welches es Entwicklern erleichtern soll solche Web Services zu implementieren. Diese Ausarbeitung betrachtet Apache CXF näher, stellt dessen Funktionalitäten dar und soll zeigen, ob es den Entwickler damit ermöglicht wird, auf einfachen Wege Web Services zu erstellen.

6.2 Allgemeines

Bei Apache CXF handelt es sich um ein so genanntes Open Source Web-Service Framework welches auf Java basiert. Es stellt dem Entwickler einen Programmier-Rahmen bereit, der es ihm erlaubt verschiedenen Schnittstellen, Verfahren und Anwendungen in Bezug auf Web Services mit geringem Aufwand zu implementieren. Solche Web Services sind XML-basierte Kommunikationsschnittellen für Verteilte Systeme, die plattformübergreifende Anwendungen realisieren sollen. CXF unterstützt zudem eine Vielzahl von Frontendprogrammiermodellen. Apache CXF wird als Open Source Software von der Apache Software Foundation unter der Apache License 2.0 veröffentlicht.[1]

Apache CXF entwickelte sich aus den zwei Webservice-Frameworks *Codhouse XFire* und *IONA Celtix* welche zu Apache CXF verschmolzen und so die Abkürzung CXF für "**C**elti**XF**ire"definierten. Die Entwickler von Celtix arbeiteten bis 2006, die von XFire noch bis Mai 2007 eigenständig an ihren Softwarelösungen und begannen danach mit der Implementierung von Apache CXF. Beide Softwarelösungen brachten eine große Zahl von Protokollen und Programmierverfahren zur Erstellung von verteilten Anwendungen in das neue Programm ein. Die Firma IONA, die vormals Celtix entwickelte, vertreibt aktuell eine Enterprise-Version von CXF mit Support- und Serviceverträgen.[2] [3]

6.3 Implementierte Protokolle

Obwohl Apache CXF im Vergleich zu anderen Framworks wie etwa Apache Axis recht schlank ist, verfügt es dennoch über eine vielfältige Auswahl von implementierten Protokolle und Verfahren zur Entwicklung von Web Services.[4] Die Entwickler erweitern zudem mit dem Voranschreiten des Projekts dieses mit zusätzlichen Ansätzen. So kann Apache CXF als ein kleines Sammelsurium von vielen, aber dennoch ausgewählten Web Service-Implementierungsverfahren verstanden werden. Es obliegt allerdings oft der Aufgabe des Entwicklers sich für das optimale Verfahren zu entscheiden. Einige Vorkenntnisse über Web Services sind somit notwendig um die optimale Schnittstelle zu spezifizieren, wenn diese nicht vom Verteilten System definiert sind. Im Folgenden werden einige der wichtigsten enthaltenen Protokolle zur Realisierung von Web Service vorgestellt.

6.3.1 SOAP

SOAP (ursprünglich für Simple Object Access Protocol) stellt ein Protokoll der Anwendungsschicht dar. Primär in Kombination mit HTTP und TCP verwendet, handelt es sich hierbei um eine Rahmenvorgabe von XML-basierten Nachrichten zur Kommunikation von Systemen. Es ermöglicht verteilten Systemen Informationen via HTTP auszutauschen und stellt somit eine wichtige Grundlage für die Entwicklung von Web Services dar.[5] SOAP war bereits in den Frameworks Codhouse XFire und IONA Celtix enthalten auf denen CXF basiert und ist nicht nur deshalb immer noch ein wichtiges Protokoll in Apache CXF zur Implementierung von Web Services.

6.3.2 WSDL

Bei WSDL (Web Services Description Language) handelt es sich um eine XML-basierte Metasprache um Web Service-Funktionen anzubieten und auszulesen. So kann vom Client der Funktionsumfang der Serveranwendungen ermittelt werden um verteilte Anwendungen zu strukturieren. WSDL wird in der Regel mit SOAP verwendet um Web Services zu realisieren. Die XML-Struktur von WSDL wird dabei durch Folgende 6 Elemente beschrieben.[6]

- **Datentypen** Daten die den Nachrichtenaustausch beschreiben
- **Nachrichten** beschreiben gesendete und empfangene Daten
- **Schnittstellen** beschreiben Dienstoperationen
- **Protokoll** beschreibt Protokolle und Datenformate einer Schnittstelle
- **Port** beschreibt Adressen für ein Protokoll
- **Service** fasst mehrere Ports zusammen

Zukünftig ist die Implementierung von WSDL 2.0 in CXF geplant, welches einige Erweiterungen und Spezifikationen für die dahinterstehende Metasprache liefern soll.

6.3.3 WS-* Spezifikationen

Eine Erweiterung des SOAP/WSDL-Ansatzes zum erstellen von Web Services, stellen die so genannten WS-* Spezifikationen des World Wide Web Consortium (W3C) dar. Diese basieren auf SOAP und WSDL, spezifizieren aber weitere, sehr spezielle Verfahren zur Abstimmung und Kommunikation von verteilten Systemen zur Implementierung bzw. Nutzung von Web Services.

Eine Vielzahl der Protokolle wurde bereits in Apache CXF implementiert, so dass diese vom Entwickler eines Web Services ohne viel Aufwand implementiert werden können. Dabei sind aktuell folgende Standards umgesetzt (Version 2.3.2, Stand 31.01.2011)[7]:

- **WS-Adressing** beschreibt Verfahren zum Austausch von Adressinformationen
- **WS Policy** beschreibt Verfahren zum Austausch von Richtlinien, Qualität, Version des Systems
- **WS-Reliable Messaging** beschreibt Verfahren, welches unter Zuhilfenahme einer Middelware sicherstellt, dass gesendete Nachrichten trotz Teilversagen des Systems zugestellt werden
- **WS-Security** beschreibt Kommunikationsprotokolle um Sicherheitsaspekte gewährleisten zu können
- **WS-SecurityPolicy** beschreibt Verfahren zur Zusicherungen an Sicherheit-Aspekten
- **WS-SecureConversation** beschreibt Verfahren zur Realisierung einer sicheren Kommunikation
- **WS-Trust** beschreibt Verfahren zum Domänen-übergreifenden Austausch von Informationen

Neben den bereits implementierten WS-* Spezifikationen sind folgende für zukünftige Versionen von Apache CXF geplant:

- **WS-Coordination** beschreibt Verfahren um Verteilte Anwendung miteinander zu koordinieren
- **WS-Atomic Transactions** erweitert WS-Coordination um Spezifikationen für kurz laufende Anwendungen
- **WS-BusinessActivity** erweitert WS-Coordination um Spezifikationen für länger laufende Anwendungen
- **WS-MetaDataExchange** beschreibt Verfahren zur Suche von Web Service-Metadaten
- **WS-Eventing** beschreibt Verfahren um ein Event-Management für Web Services zu realisieren
- **WS-Transfer** beschreibt Verfahren um Übertragungsspezifikationen für Web Services zu definieren

6.3.4 Weitere Protokolle

Neben den bereits vorgestellten Protokollen zur Realisierung von Web Services, sind in Apache CXF noch viele weitere Ansätze in CXF implementiert. Dazu zählen unter andrem COBRA, Pure XML und REST. Eine aktuelle und vollständige Liste findet sich auf der Homepage des Projektes. (siehe [7])

6.4 Implementierte Programmierverfahren

Neben den vorgestellten Web Service Protokollen verfügt CXF über eine Auswahl von implementierten Programmierverfahren die überwiegend auch zur Frontendprogrammierung bei der Entwicklung von Web Services genutzt werden können. Im Folgenden wird eine Auswahl der wichtigsten Verfahren vorgestellt.

6.4.1 Spring

Bei Spring handelt es sich um ein eigenständiges Open Source Framework auf Java-Basis. Spring vereinfacht Programmierlogiken und soll gute Programmierpraktiken fördern. Mit Spring soll es vereinfacht werden die Applikationskomponenten eines Programmes auszulagern. Die Besonderheit, die Spring als eine wichtige Komponente von CXF auszeichnet, ist die Tatsache, dass viele verschiedene Plattformen und Systeme unterstützt werden. Spring wurde in Apache CXF integriert, da mittels dieses Frameworks eine gute Möglichkeit besteht Verteilte Anwendungen zu realisieren. Ein Beispiel zur Implementierung von Spring findet sich im Kapitel *Beispiel.*(s.u.)[8]

6.4.2 JAX-WS

Bei der **J**ava **A**PI for **X**ML - **W**eb **S**ervices handelt es sich um eine Java-Programmschnittstelle zum Erstellen von Web Services. Es erleichtert den Programmierer die Einrichtung des Web Services und der Kommunikation zwischen verteilten Anwendungen bzw. Systemen. JAX-WS integriert die Protokolle SOAP, WSDL sowie einige WS-* Standards, weshalb sich diese API gut in Apache CXF integrieren lässt.[9]

6.4.3 JAX-RS

Die **J**ava **A**PI for **R**ESTful Web **S**ervices ist eine Java-Programmierschnittstelle zum Erstellen eines Web Services unter Verwendung des Software-Architekturstils *Representational State Transfer* (kurz REST).

Ebenso wie bei JAX-WS erleichtert JAX RS es den Programmierer Web Services zu implementieren und verteilte Anwendungen zu realisieren. JAX-RS wurde im Rahmen des Java Community Process von einem Konsortium um die Firma Sun Microsystems erarbeitet. Die Integrierung in CXF erlaubt dem Entwickler abseits von SOAP/WSDL zu arbeiten.[10]

6.4.4 Weiter Programmierverfahren

Neben den vorgestellten Programmieransätzen hat Apache CXF weitere Verfahren integriert und wird mit Voranschreiten des Projektes um weitere Möglichkeiten zur Implementierung von Web Services erweitert. Einige weiter Ansätze und Programmierschnittstellen sind Java Business Integration (JBI), Java EE Connector Architecture (JCA) und Java Management Extensions (JMX). Eine vollständige Liste der aktuell implementierten und geplanten Programmieransätze findet sich auf der Homepage des Projektes. (siehe [7])

6.5 Beispiel

Im Folgenden soll an einem einfachen Beispiel gezeigt werden wie ein Projekt mit Hilfe von CXF erstellt werden kann. Es wird ein trivialer Service mithilfe von Spring und Zuhilfenahme von WSDL erstellt. Der Code kann mit Hilfe jeder IDE erstellt werden, die beste Unterstützung von Seiten Apache CXF erhält man allerdings für Eclipse. (für genauere Anweisungen siehe hier: [11]) Andere IDE wie Netbeans oder das Kompilieren mit Hilfe von Apache Maven und Apache Ant ist aber ebenso möglich. Am stabilsten läuft CXF auf der JDK 1.5, allerdings wird auch die neuere 1.6 unterstützt.

Vor der eigentlichen Implementierung muss ein neues Projekt angelegt werden und einige Dateien eingebunden werden. Folgende JAR-Files sind Standard CXF-Bibliotheken die für die meisten Projekte benötigt werden.

```
commons-logging-1.1.jar
geronimo-activation_1.1_spec-1.0-M1.jar
geronimo-annotation_1.0_spec-1.1.jar
geronimo-javamail_1.4_spec-1.0-M1.jar
geronimo-servlet_2.5_spec-1.1-M1.jar
geronimo-ws-metadata_2.0_spec-1.1.1.jar
jaxb-api-2.0.jar
jaxb-impl-2.0.5.jar
jaxws-api-2.0.jar
neethi-2.0.jar
saaj-api-1.3.jar
```

```
saaj-impl-1.3.jar
stax-api-1.0.1.jar
wsdl4j-1.6.1.jar
wstx-asl-3.2.1.jar
XmlSchema-1.2.jar
xml-resolver-1.2.jar
```

Listing 6.1: CXF-Projekt - allgemeine JAR-Dateien

Um Spring und CXF (Versionsangabe) verwenden zu können müssen zusätzlich folgende Dateien eingebunden werden.

```
aopalliance-1.0.jar
spring-core-2.0.8.jar
spring-beans-2.0.8.jar
spring-context-2.0.8.jar
spring-web-2.0.8.jar
cxf-2.1.jar
```

Listing 6.2: CXF-Projekt - Spring und CFX JAR-Dateien

Sind alle Dateien in das Projekt eingebunden, kann mit der Implementierung des Service begonnen werden. Für die Erstellung eines Web Services müssen eine Interface- und eine Implementierung-Datei angelegt werden. Die Interface-Datei sieht im aktuellen Beispiel wie folgt aus.

```
package demo.spring;
import javax.jws.WebService;
@WebService
public interface HelloWorld {
     String sayHi(String text);}
```

Listing 6.3: CXF-Projekt - Interface-Datei

Die zugehörige Implementation enthält folgenden Code.

```
package demo.spring;
import javax.jws.WebService;
@WebService(endpointInterface = "demo.spring.HelloWorld")
public class HelloWorldImpl implements HelloWorld {
     public String sayHi(String text) {
          System.out.println("sayHi called");
          return "Hello " + text; }}
```

Listing 6.4: CXF-Projekt - Implementation-Datei

Der Service ist damit grundlegend vollständig um eine *HalloWorld*-Antwort zu produzieren. Durch den Befehl *@WebService* kann CXF automatisch eine entsprechende WSDL-Datei erstellen. Um den Web Service mittels Spring nutzen zu können müssen noch die Server Beans angepasst werden. Dazu müssen im *WEB-INF*-Ordner die Datei *beans.xml* folgendermaßen anpassen.

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jaxws="http://cxf.apache.org/jaxws"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd">
  <import resource="classpath:META-INF/cxf/cxf.xml" />
  <import resource="classpath:META-INF/cxf/cxf-extension-soap.xml" />
  <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />
  <jaxws:endpoint id="helloWorld"
    implementor="demo.spring.HelloWorldImpl"
    address="/HelloWorld" />
</beans>
```

Listing 6.5: CXF-Projektt - beans.xml

Um den Spring-Service zu komplettieren muss zuletzt die web.xml-Datei angepasst werden.

```
<web-app><context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>WEB-INF/beans.xml</param-value>
  </context-param>
  <listener><listener-class>
      org.springframework.web.context.ContextLoaderListener
  </listener-class></listener>
  <servlet><servlet-name>CXFServlet</servlet-name>
          <display-name>CXF Servlet</display-name>
    <servlet-class>org.apache.cxf.transport.servlet.CXFServlet
    </servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>
  <servlet-mapping><servlet-name>CXFServlet</servlet-name>
                  <url-pattern>/ *</url-pattern>
  </servlet-mapping>
</web-app>
```

Listing 6.6: CXF-Projekt - web.xml

Mit dem in der web.xml definierten ContextLoaderListener wird das Spring-Framework entsprechend gestartet und die beans.xml geladen. Damit ist der Spring-Web Service vollständig implementiert und einsatzbereit. Um den Service testen zu können ist abschließend die Implementierung eines Clients notwendig. Dies ist allerdings recht einfach und sieht im aktuellen Beispiel wie folgt aus.

```
package demo.spring.client;
import
    org.springframework.context.support.ClassPathXmlApplicationContext;
import demo.spring.HelloWorld;
public final class Client { private Client() { }
    public static void main(String args[]) throws Exception {
        ClassPathXmlApplicationContext context
            = new ClassPathXmlApplicationContext(new String[]
                {"demo/spring/client/client-beans.xml"});
        HelloWorld client = (HelloWorld)context.getBean("client");
        String response = client.sayHi("Joe");
        System.out.println("Response: " + response);
        System.exit(0); }}
```

Listing 6.7: CXF-Projekt - Client

Abschließend muss noch die bean-Datei des Servers angepasst werden. Im aktuellen trivialen Beispiel einfach an den entsprechenden Localhost-Port.

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jaxws="http://cxf.apache.org/jaxws"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
   http://www.springframework.org/schema/beans/spring-beans.xsd
 http://cxf.apache.org/jaxws http://cxf.apache.org/schemas/jaxws.xsd">
  <import resource="classpath:META-INF/cxf/cxf.xml" />
  <import resource="classpath:META-INF/cxf/cxf-extension-soap.xml" />
  <import resource="classpath:META-INF/cxf/cxf-servlet.xml" />
  <jaxws:endpoint id="helloWorld"
    implementor="demo.spring.HelloWorldImpl"
    address="/HelloWorld" />
</beans>
```

Listing 6.8: CXF-Projekt - Client beans.xml

Als Ergebnis haben wir einen Client und einen Server zur Realisierung eines trivialen Web Services, der problemlos erweitert werden kann.[12]

6.6 Anwenderfreundlichkeit

Apache CXF ist im Vergleich zu vielen anderen Open Source-Projekten recht weit entwickelt. Es werden regelmäßig Updates und Weiterentwicklungen veröffentlicht, sowie Fehler behoben. Trotz der schlankeren Implementierung ist das Projekt vielfältig und hat viele Ansätze zur Entwicklung und Gestaltung von Web Services integriert. Dieser Punkt kann aber dahingehend als Nachteil betrachtet werden, dass eine große Vielfalt an Entwicklungsmöglichkeiten viele Vorkenntnisse beim Entwickler voraussetzt. Das Projekt liefert nur eine spärliche Dokumentation seiner Funktionalitäten, was dazu führt, dass auch professionelle Entwickler eine gewisse Einarbeitungszeit in das Projekt investieren müssen, um damit produktiv arbeiten zu können. User die über geringere Programmierkenntnisse verfügen und Apache CXF als Instrument verwenden wollen, um auf einfachen Wege Web Services zu implementieren, werden eine längere Einarbeitungszeit benötigen. Aufgrund der vielfältigen Implementierung und der entgegenstehenden schlechten Dokumentation ist es für User ohne großes Hintergrundwissen nahezu unmöglich die Funktionalitäten von CXF voll auszunutzen und die Vorzüge gegenüber anderen Entwicklungsmethoden wahrzunehmen.

6.7 Fazit

Apache CXF ist ein vielfältiges, aber dennoch schlankes Web Service Framework, welches dem Entwickler bei der Implementierung von Web Service unterstützen soll. Das Projekt wird seit mehreren Jahren kontinuierlich weiterentwickelt und hat deshalb sehr viele implementiere Schnittstellen, Protokolle und Verfahren. CXF kann somit als kleines Sammelsurium an Methoden zum Erstellen von Web Services verstanden werden. Dennoch ist es in mancher Hinsicht beschränkter als andere Frameworks wie etwa Apache Axis. Es liefert aber auch Funktionalitäten, wie die Integierung von Spring, welches es gegenüber von anderen Frameworks abhebt. Zu kritisieren ist die lückenhafte Dokumentation, welche es Entwicklern mit geringeren Vorkenntnissen erschwert sich schnell in das Projekt einzuarbeiten und die Funktionalitäten im vollen Umfang einzusetzen. Für eine wirtschaftliche bzw. kommerzielle Nutzung müsste eine bessere Dokumentation und Nutzerunterstützung angeboten werden. Dennoch ist Apache CXF ein gut entwickeltes Framework, welches nach einer gewissen Einarbeitungszeit effektiv zur Erstellung von Web Services genutzt werden kann.

Literaturverzeichnis

[1] Apache Software Foundation (Hrsg.) *Apache CXF: An Open-Source Services Framework* http://cxf.apache.org 21.01.2011

[2] Codhouse (Hrsg.) *Codehouse XFire* http://xfire.codehaus.org 03.01.2011

[3] IONA (Hrsg.) *Celtix: The Open Source Java ESB* http://celtix.ow2.org 03.01.2011

[4] Thomas Bayer *Apache Axis2, CXF und Sun JAX-WS RI im Vergleich* http://www.predic8.de/axis2-cxf-jax-ws-vergleich.htm 04.01.2011

[5] W3school (Hrsg.) *SOAP Tutorial* http://www.w3schools.com/soap/default.asp 05.01.2011

[6] W3school (Hrsg.) *WSDL Tutorial* http://www.w3schools.com/wsdl/default.asp 05.01.2011

[7] Apache Software Foundation (Hrsg.) *Apache CXF - Projekt Status* http://cxf.apache.org/project-status.html 31.1.2011

[8] SpingSource (Hrsg.) *The Standard for Enterprise Java Development* http://www.springsource.com/developer/spring 05.01.2011

[9] Oracle Corporation (Hrsg.) *Getting Started with JAX-WS Web Services* http://netbeans.org/kb/docs/websvc/jax-ws.html 21.01.2011

[10] Oracle Corporation (Hrsg.) *Jersey: RESTful Web services made easy* http://wikis.sun.com/display/Jersey/Main 21.01.2011

[11] Apache Software Foundation (Hrsg.) *Apache CXF - Setting up Eclipse* http://cxf.apache.org/setting-up-eclipse.html 05.01.2011

[12] Apache Software Foundation (Hrsg.) *Apache CXF - Writing a service with Spring* http://cxf.apache.org/docs/writing-a-service-with-spring.html 05.01.2011