Dokumentation des Web Service-Clients für den MAN WebService

Rainer Sawitzki
Javacream
Version 3.0
30.10.2017

Konventionen dieses Dokuments

- Links auf externe Dokumente werden *Courier kursiv unterstrichen* dargestellt.
- Verzeichnisnamen und Dateinamen werden *Courier kursiv* dargestellt.
- Code-Beispiele werden in `Courier-Schriftart` formatiert.
- Abkürzungen werden in `COURIER-GRO□BUCHSTABEN` formatiert.
- Die folgende Tabelle enthält die im Text benutzten Abkürzungen und Begriffe:

WSDL	Eine Schnittstellenbeschreibung im Schema der Web Service Description Language
XSD	Ein XML-Schema-Dokument beschreibt die Struktur eines daraus erzeugten Dokuments.
SOAP	Ein XML-Dokument, das zum Datenaustausch zwischen einem Web Service Consumer und einem Web Service PROVIDER ausgetauscht wird. Ein SOAP-Envelope besteht aus einem Header- und einem Body-Bereich
PAYLOAD	Die Daten, die ein einem SOAP-Body transportiert werden
PROVIDER	Eine Web Service-Implementierung, die auf einem Server zur Verfügung gestellt wird
CONSUMER	Ein Web Service-Client, der die Operationen eines PROVIDERs aufruft
POM	Ein Projekt-Deskriptor, der einen Maven-basierten Build-Prozess für ein JAva-Projekt definiert.
JAXB	Das Java API for XML Binding definiert ein Framework zum Mapping von Java-Typen nach XML-Schema und umgekehrt. Weiterhin enthält JAXB eine Generator.
SPRING	Ein Context und Dependency-Injection Framework, *https://de.wikipedia.org/wiki/Spring_(Framework)*
ECLIPSE	Eine Java-Entwicklungsumgebung
MAVEN	Apache Maven ist ein Buildwerkzeug, siehe *https://de.wikipedia.org/wiki/Apache_Maven*
SPRING BOOT	Eine Erweiterung von SPRING, die beispielsweise einen ausgefeilten Build-Prozess definiert,

	https://de.wikipedia.org/wiki/Spring_(Framework)
ENDPOINT	Die Adresse eines PROVIDERs, in der Regel eine http-URL
KEYSTORE	Eine Datei, die Security-relevante Informationen wie Schlüssel und Zertifikate enthält.

Hintergrund

Die von MAN gelieferte WSDL definiert einen generischen Web Service, der nur aus einer einzigen Operation invoke besteht. Die Struktur der für diese Operation zu übermittelnden Daten, der PAYLOAD, ist nicht Bestandteil der WSDL.
MAN lieferte weiterhin durch Beispiel-Requests Informationen für den Aufruf einzelner Operationen. Darin enthalten sind die einzelnen zu übermittelnden Daten und daraus kann die notwendige Struktur ermittelt werden.
Weiterhin verlangt die Kommunikation mit dem Web Service eine Authentifizierung über ein vom Aufrufer zu sendendes Zertifikat.
Der PROVIDER des Web Services wird von MAN zur Verfügung gestellt und betrieben. Die ASC AG möchte als CONSUMER auf diesen Web Service zugreifen.

Vorarbeiten

Herr Andreas Becker hat aus den Beispiel-Requests eine Schema-Datei, *EOI-Server.xsd*, erzeugt, die die Payload-Struktur definiert. Weiterhin wurde von Ihm versucht, den CONSUMER aus der bekannten WSDL sowie der erstellten XSD zu erzeugen.

Analyse

Der Versuch der automatischen Generierung des Clients hat so leider nicht funktioniert. Insbesondere die von MAN gelieferten Beispiele für die SOAP-Anfragen verhindern einen einfachen Einsatz des Code-Generators.

Konzeption

Auf Grund des von MAN gewählten generischen Ansatzes für die WSDL wird auch für den CONSUMER eine generische Lösung bevorzugt. Um die Anwendungsentwicklung zu vereinfachen und zu standardisieren wurde folgendes Konzept erstellt:

- In der CONSUMER-Implementierung werden Java-Typen benutzt, die der Schema-Definition der identifizierten PAYLOAD-Struktur entsprechen.

- Diese Java-Typen werden mit JAXB aus der XSD erzeugt.
- Für den eigentlichen Aufruf des PROVIDERs wird das SPRING-Framework mit SPRING BOOT benutzt. Dieses Framework ist bei der ASC AG etabliert und vereinfacht die notwendigen Programmsequenzen der Implementierung drastisch.
- Zur Realisierung werden ECLIPSE-Projekt benutzt. Diese Entwicklungsumgebung ist bei der ASC-AG etabliert.
- Als Build-Werkzeug wird MAVEN benutzt. Dieses Werkzeug ist bei der ASC-AG etabliert.

Umsetzung

Zur Umsetzung wurden zwei ECLIPSE-Projekte angelegt:

- org.javacream.asc_ag.types
  - Ein einfaches MAVEN-Projekt
  - Erzeugt die Java-Typen aus der EOI_Server.xsd
- org.javacream.asc_ag.ws
  - Ein SPRING BOOT-Projekt
  - Darin wird der WebServiceClient definiert, der als CONSUMER die Aufrufe zum PROVIDER durchführt. In dieser Implementierung wird als Hilfsklasse das WebServiceTemplate von SPRING benutzt.
  - Die Konfiguration der SPRING BOOT-Anwendung erfolgt in der application.properties sowie in der WebServiceClientConfiguration.
  - Die Konfiguration der application.properties umfasst:
    - client.default-uri
      - ENDPOINT-Adresse des PROVIDERs
    - client.ssl.keystore.path
      - Die KEYSTORE-Datei, die das Client-Zertifikat enthält
    - client.ssl.keystore.password
      - Das Password für die KEYSTORE-Datei
    - client.ssl.keystore.type
      - Der Typ des KEYSTOREs, aktuell pkcs12
    - jaxb.package
      - Das Haupt-Paket, in dem JAXB nach Java-Datentypen mit XML-Mappings sucht
  - Die Konfiguration durch die WebServiceClientConfiguration umfasst:
    - Einen Jaxb2Marshaller, der das ${jaxb.package} scannt
    - Ein WebServiceTemplate, das die Endpoint-Adresse benutzt sowie den Jaxb2Marshaller zum Marshalling und Unmarshalling benutzt
  - Die PAYLOAD-Struktur wird durch die Klasse DmsHeaderAndOrders05 definiert.
  - Die eigentliche Implementierung WebServiceClient enthält nur eine einzige Methode, die an das WebServiceTemplate delegiert.

    - Weiterhin enthält das Projekt einen Testfall, den `WebServiceClientTest`.

Hinweise

- Die von MAN gewählte Struktur des `SOAP`-Bodies mit zwei Elementen widerspricht etwas dem Standard von Web Services. Die Implementierung ist somit stark XML-lastig.
- Die Implementierung ist nicht bzw. nur schwer generisch möglich.
- Die Fehlerbehandlung ist noch nicht implementiert.
- Zur Ausgabe der `SOAP`-Requests und `SOAP`-Responses wird ein `ClientInterceptor` benutzt.

Verwendung

- Der `WebServiceClient` wird in das Projekt kopiert, das als `CONSUMER` auftreten soll. Weiterhin muss die Konfiguration des `CONSUMER`-Projekts die in `org.javacream.asc_ag.ws` enthaltenen Einstellungen übernehmen.
- Das `CONSUMER`-Projekt hat eine Abhängigkeit auf das Projekt `org.javacream.asc_ag.ws`.
- `org.javacream.asc_ag.ws` kann selbstverständlich umbenannt werden. Wichtig ist dann aber die Anpassung der Property `${jaxb.package}`.

Verwendung in einem Plex-Projekt

Falls die Anwendung in einem Nicht-Maven-basierten Projekt eingesetzt werden soll, müssen alle Spring-Boot-Dependencies sowie die generierten `JAXB`-Klassen händisch in den Klassenpfad übernommen werden. Anschließend muss der Spring-Kontext aus den eigenen Programmsequenzen heraus gestartet werden. Ein Beispiel ist in dem Eclipse-Projekt `PlainOldJava` gegeben. Zum Zugriff auf den WebServiceClient wird einfach die Methode `WebServiceClientHelper.getWebServiceClient()` aufgerufen.

Ablage

Die Projekte sowie diese Dokumentation sind in aktueller Version unter dem Git-Repository *https://github.com/Javacream-Projects/org.javacream.asc_ag* zu finden.