Erfahrungsbericht: Webservice Entwicklung mit Oracle ADF

Florian Schulze und Christian Feicke
Computacenter AG & Co oHG
Mariendorfer Damm 1-3, 12099 Berlin

Schlüsselworte:

Application Development Framework (ADF), JDeveloper 10g/11g, Webservices, Testing, Build Management, Erfahrungsbericht

Einleitung

Die Erstellung von Webservices wird seit geraumer Zeit vom Oracle JDeveloper unterstützt. Wie so oft unterscheidet sich Theorie und Praxis. Mit diesem Vortrag soll ein Erfahrungsbericht gegeben werden, der von typischen Stolperfallen aus unseren Projekten berichtet, Hinweise und BestPractices zu ADF Projekten im WebserviceUmfeld geben will und damit für die Planung von WebService Projekten von Bedeutung sein kann. Dabei wird auch auf die Unterschiede zwischen dem immer noch weit verbreiteten ADF 10g und dem aktuellen ADF 11g eingegangen und es werden Empfehlungen für projektbegleitende Tools und Vorgehensweisen gegeben.

Alles neu macht der Mai – kurz die wesentlichen Unterschiede zwischen JDeveloper 10g und 11g im Bereich Webservices

Beim neuen JDeveloper 11g hat sich gegenüber seinem Vorgänger JDeveloper 10g (10.1.3.x) unter der Haube einiges getan. Der vormals verwendete OC4J wurde durch einen Weblogic Server ersetzt. Der Wechsel der RPC-orientierten Webservice Runtime JAX-RPC auf den Nachrichten-orientierten Nachfolger JAX-WS bringt einen Paradigmen-Wechsel mit sich: erstmals können asynchrone Webservice-Aufrufe stattfinden, bei der bewusst zeitverzögerte oder ausbleibende Antworten erlaubt sind. Musste Oracle JAX-RPC durch proprietäre Erweiterungen an die regen Spezifikations-Erweiterungen im Webservice-Umfeld anpassen, so wird bei JAX-WS weitestgehend die Referenzimplementierung verwendet. Ein großer Gewinn ist, dass der Object-XML-Mapper nun alle Datentypen der XML-Schema Spezifikation sicher abbildet. Es ist nicht mehr notwendig, Mapper für die Serialisierung komplexer Objekte in die XML-Repräsentation selber zu erzeugen. Auf Basis dieser XML-Schemas können die Nachrichten vollständig validiert werden. Dank JAX-WS braucht nur noch die Annotation `@ValidateSchema` gesetzt werden

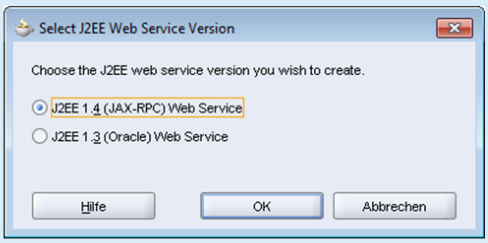
JDeveloper 10g	JDeveloper 11g
Java API for XML-based RPC (JAX-RPC) 1.1	Java API for XML-based RPC (JAX-RPC) 1.1 Java API for XML-Based Web Services (JAX-WS) 2.1.4
Oracle Container For J2EE (OC4J) 10g	Oracle Weblogic Server 10.3 (WLS)
	

Abb. 1: Unterschied Webservice Support 10g und 11g

Was bringt der Umstieg auf JDeveloper11g sonst noch mit sich?
Die programmatische Erstellung von Webservices hat sich mit der Version 11g durch die Unterstützung im Bereich der Annotationen deutlich verbessert. Wo früher ein Wizard (und die Maus) bemüht werden musste, um aus Java-Klassen Webservice zu generieren, reicht heute das Setzen von wenigen Annotationen aus. Mit der Annotation bestimmt man, ob ein JAX-WS oder JAX-RS aka. Restful Webservice zum Einsatz kommen soll. Das Projekt-Setup wird implizit zu einem Webprojekt umgewandelt und benötigte Dateien werden generiert.

Wo beim JDeveloper10g das XML-Schema einer WSDL-Datei separat bearbeitet werden musste, erlaubt der WSDL-Editor im JDeveloper11g direktes Bearbeiten des enthaltenen XML-Schemas. Dadurch können Datentypen des Schemas bequem aus dem WSDL-Editor heraus mit Restriktionen und den beschränkenden Facetten versehen werden. Diese dienen als Grundlage einer Schema-Validierung des SOAP-Body gegen die definierten Datentypen des verwendeten XML-Schemas.

Zum Thema Migration /Kompatibilität: Eine gute und eine schlechte Nachricht…

Die Gute: JAX-RPC und JAX-WS reden miteinander. JAX-RPC versteht die SOAP-Nachrichten eines JAX-WS Webservice und umgekehrt.

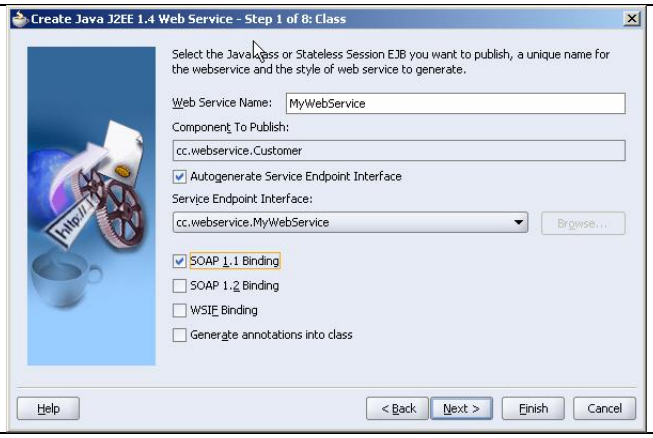
Die schlechte Nachricht: Bei einer Migration von JDeveloper 10g und JAX-RPC auf JDeveloper 11g und JAX-WS müssen neben den Bibliotheken (und ihren Referenzen) auch selbst erstellte Message Handler geändert werden. Diese kommen beispielsweise zum Einsatz, wenn eigene XML-Mapper verwendet wurden oder die Nutzlast einer Nachricht gegen das XML-Schema validiert werden sollte. Wo vorher SAAJ1.2 und die proprietärer Oracle Erweiterung (ORASAAJ) verwendet wurde, kommt jetzt SAAJ1.3 mit veränderter API zum Einsatz.

Die Alternative zum Wizard „Create Webservice“ oder Wie generiere ich automatisiert die Services?

Der Webservice Assembler (WSA) im JDeveloper ist das zentrale Tool rund um die Webservice Erstellung und Klassengenerierung. Es ist u.a. das Werkzeug für die Generierung der WSDL.

Der WSA liegt in zwei Formen vor – als GUI Version im JDeveloper und als Kommandozeilenversion.

GUI Version	Commandline Version

	`java -jar wsa.jar -help`
	`Verwendung:`
	`java -jar wsa.jar -<command> -debug -help`
	`M÷gliche WHERE-Befehle:`
	`analyze`
	`aqAssemble`
	`assemble`
	`corbaAssemble`
	`dbJavaAssemble`
	`ejbAssemble`
	`fetchWsdl`
	`genApplicationDescriptor`
	…

Die Kommandozeilenversion verfügt über einen gegenüber der GUI Version erweiterten Funktionsumfang. Durch die Möglichkeit, Services über die Kommandozeile zu generieren, kann dieser Prozess automatisiert in Build-Läufe integriert werden. Automatisierung hat neben dem Zeitgewinn noch den wichtigen Aspekt der Nachvollziehbarkeit. Im Sinne der Qualitätssicherung ist dieser Aspekt in Projekten nicht zu unterschätzen.

In unseren Projekten sind wir auf Probleme mit dem wsa im Zusammenhang mit Restriktionen auf den Feldern gestoßen. Von anderen Dienstleistern gelieferte Schemas mit Restriktionen liefen in die Warnmeldung:

```
WARNUNG: OWS-00102: …. Grund f³r die Nichterstellung eines
benutzerdefinierten Java-Typs: Verwendung eines nicht implementierten
Features
```

Die Warnmeldung führt jedoch zu einem Abbruch und Nichtgenerierung der Beans. Ein Beispiel für eine solche Restriktion ist der Abbildung zu entnehmen:

```
<xsd:element name="Seitenzahl" minOccurs="0" maxOccurs="1">
    <xsd:simpleType>
        <xsd:restriction base="xsd:long">
            <xsd:minExclusive value="0"/>
            <xsd:maxInclusive value="9999999999"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

Abb. 2 – Problem Schemarestriktion (Beispiel)

Abhilfe schafft nach unser Erfahrung nur eine Umformulierung in die (fachlich gleiche) Variante:

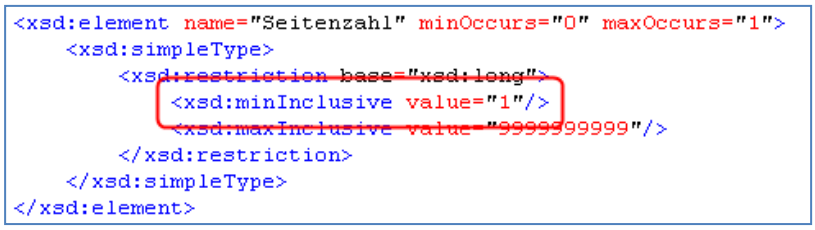


Abb. 3 – Lösung Schemarestriktion (Beispiel)

Der Umgang mit Restriktionen auf Schemaebene sollte bei der Anforderungsdefinition bzw. in der Implementierung frühzeitig beachtet werden, um nicht zu erhöhten Aufwänden zu führen, da späte Schemaänderungen womöglich zusätzliche Absprache- und Implementierungsaufwände nach sich ziehen.

Tests und Fehleranalyse

Für ein erfolgreiches Projekt bzw. Produkt sind umfangreiche Tests während der Implementierung, vor Produktivstellung sowie während des Betriebes im Rahmen des Monitorings zu planen. In den folgenden Abschnitten werden Hinweise und Tipps für erfolgreiches Testen gegeben.

Testen mit den mitgelieferten Oracle Tools
Oracle liefert Testtools für Webservices bereits mit. Damit setzt sich Oracle von Wettbewerbern ab. Im JDeveloper10g/OAS 10.1.3 sind die Möglichkeiten rudimentär, in JDeveloper11g wurden sie verbessert. Allgemein lässt sich sagen, dass die Möglichkeiten begrenzt und nicht an die Möglichkeiten spezialisierter Testtools heranreichen. Für erste Funktionstests reichen sie jedoch aus.

Testen mit soapUI
soapUI ist ein Werkzeug für funktionale Webservices Tests sowie Lasttests. Mit den Features gegenüber den mitgelieferten Oracle Testtools ist es für uns das präferierte Tool in Projekten. Es stehen zwei Versionen zur Verfügung. Eine kostenfreie, nicht in der Laufzeit beschränkte Version und eine kostenpflichtige Version mit erweitertem Funktionsumfang. soapUI steht als Standalone Programm und als Plugin zur Verfügung; es kann in Maven, NetBeans, IntelliJ IDEA, Eclipse, JBossWS/JBossIDE 2.0.0+ integriert werden. Der JDeveloper wird leider nicht unterstützt.

Bereits die Freeware stellt ein solides WebService Testing Framework zur Verfügung, welches in kleineren / wenig komplexen Webserviceprojekten eingesetzt werden kann.
Es besteht die Möglichkeit, die Tests in „Project“, „TestSuite“, „TestCase“, „TestStep“ zu strukturieren. Die SOAP Requests dafür können komfortabel von bestehenden WSDL Dateien generiert werden lassen. Die Testsuites können später auf unterschiedliche Service-Endpoints getestet werden.

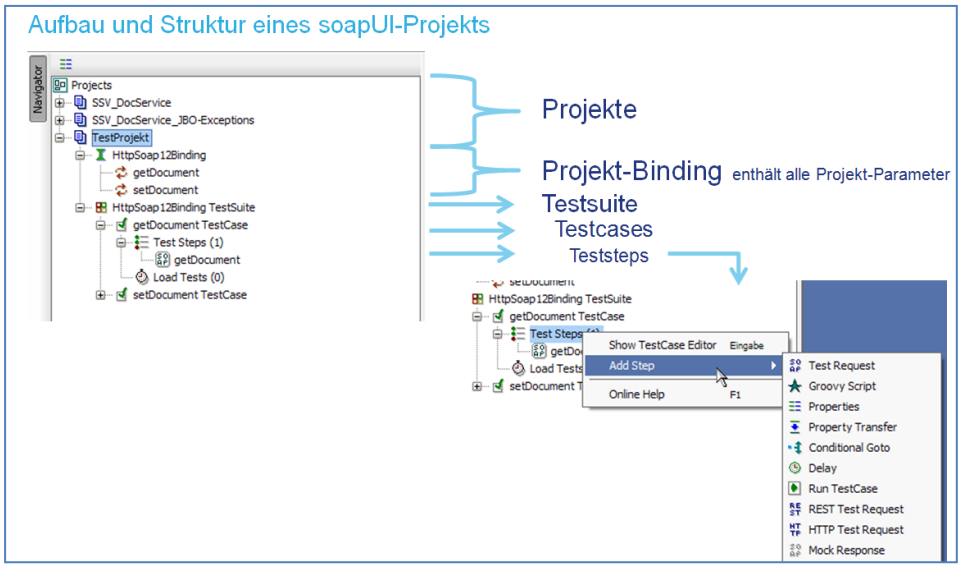


Abb. 4 – Aufbau eines soapUI Projektes

Hervorzuheben ist die Möglichkeit, Tests über ein grafisches Frontend sowie Tests automatisiert über Kommandozeilenfunktionen ablaufen zu lassen. Damit wird es unter Anderem möglich, Tests in einen automatisierten Build Prozess zu integrieren, um so beispielsweise die Anforderungen der Qualitätssicherung zu erfüllen.

Die Pro Version eignet sich, wenn Webservices komplexer und die Testcases dynamischer werden sollen. So können Testdaten (Inputs) aus zusätzlichen Quellen wie Excellisten oder über JDBC angebundene Datenbanken herangezogen werden. WSDL-Refactoring wird unterstützt und es kann ein bereits erstellter Request auf neu hinterlegte WSDL gemappt werden, d.h. der Request muss nicht vollständig neu geschrieben werden.

Fehleranalyse durch Auditing

Webservice Projekte zeichnen sich dadurch aus, dass hier verschiedene Systeme interagieren. SOAP Requests der Fremdsysteme gegen das eigene System können bereits die Ursache für fachliches oder technisches Fehlverhalten der Schnittstelle sein. Für die Fehleranalyse ist es hilfreich, die Anfrage-/Antwortnachricht auf dem empfangenden System zu loggen. Der Oracle Applicationserver stellt eine Möglichkeit bereit, die Requests und Responses mitzuschreiben (standardmäßig ist das Mitschreiben der Requests aus Performance Gründen sinnvoller Weise nicht aktiviert). Alternativ kann das Auditing auch im Deployment Deskriptor des JDeveloper-Projektes gesteuert werden. Da sich die Produktionssysteme sich in der Regel im Loggingverhalten (LogLevel) von den Entwicklungssystemen unterscheiden, ist aus unseren Erfahrungen das servergesteuerte Auditing empfehlenswert.

Folgende Schritte sind am Beispiel des OAS 10.1.3 notwendig, um das Loggen einzuschalten:

Navigieren Sie auf dem EM auf den entsprechenden Service, wählen Sie hier „Administration“ und schalten Sie das Auditing über die Schaltfläche „Enable/Disable Features“ ein

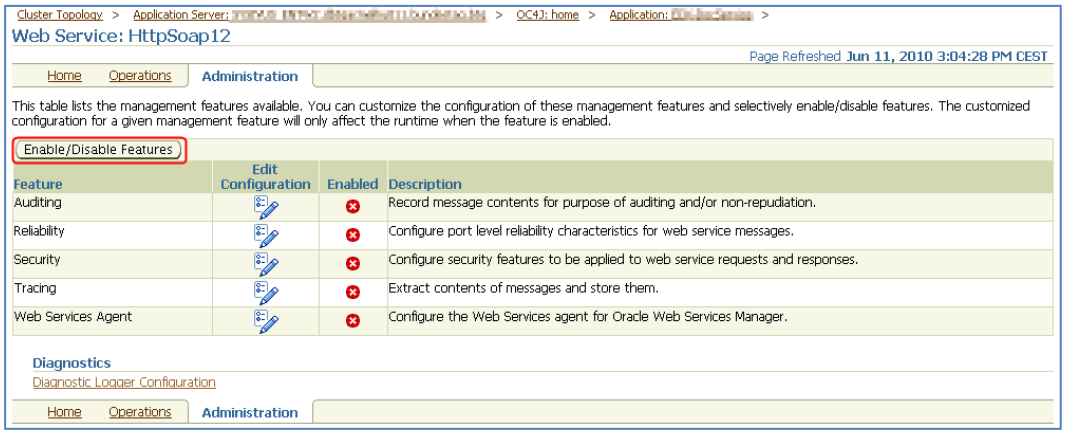


Abb. 5 – Auditing einschalten

Damit die Request /Responses von der Protokollierung erfasst werden, ist ein noch ein geeigneter LogLevel festzulegen. Das geschieht über die „Diagnostic Logger Configuration“:

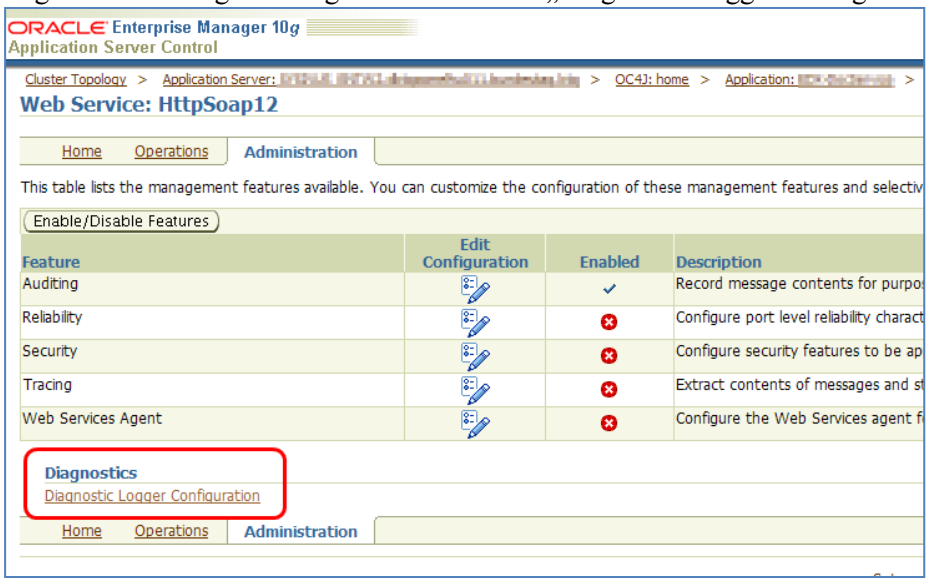


Abb. 6 – Auditing: LogLevel festlegen

Hier ist mindestens der Loglevel „FINE“ festzulegen, um das gewünschte Verhalten zu erreichen. Das Logfile log.xml (sowie dessen rollierenden Versionen logXX.xml) werden entgegen der Angabe in der Dokumentation in folgendem Verzeichnis gespeichert:

```
$ORACLE_HOME/j2ee/home/log/home_default_group_1/oc4j
```

Build Management mit Apache Ant

Die im JDeveloper zur Verfügung stehende flexible und konfigurierbare Lösung mit Deployment Profilen ist in größeren Projekten aus unserer Sicht nicht ausreichend. Um eine höhere Automatisierung im Build Prozess mit möglichst wenig Interaktion und manuellen Eingriffen des Benutzers zu erreichen, wird häufig Apache Ant eingesetzt.
Beim Einsatz der oben genannten Deployment Profile in JDeveloper 10g ist in Verbindung mit Subversion als Versionierung zu beachten, dass die .svn Verzeichnisse nicht in der Default

Ausschlussmaske stehen. Ansonsten würden Versionierungsinformationen mit ins Deployment Target geschrieben werden.

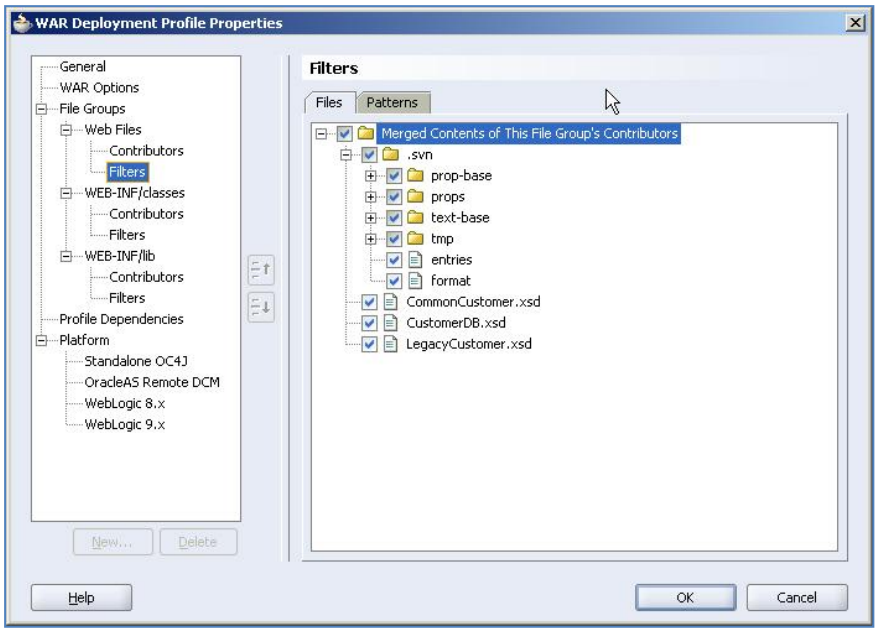


Abb. 7 - Deployment Profile mit aktivierten Subversion Informationen

Die Ausschlussmaske des Deployment-Profils lässt sich jedoch konfigurieren. Die CVS Information sind standardmäßig bereits ausgeschlossen. Beim JDeveloper 11 sind die .svn Informationen bereits ausgeschlossen.

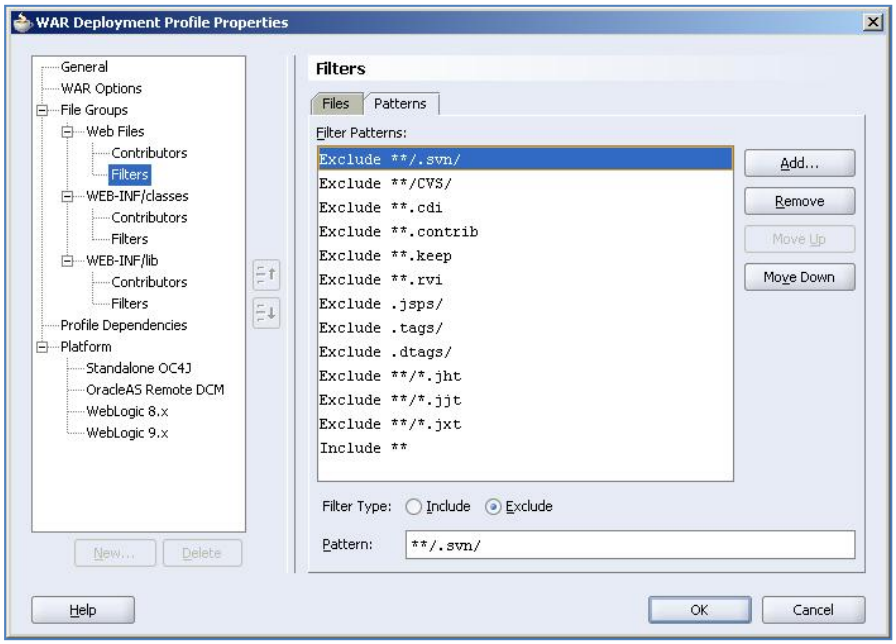


Abb. 8 - Deployment Profile mit Pattern für Subversion versehen

Fügen Sie also im JDeveloper 10g den Pattern **/.svn/ hinzu, um keine Subversion Versionierungsinformationen mit im Deployment Target zu speichern.

Mit Hilfe eines Wizards lässt sich aus den fertig konfigurierten Project Properties ein Ant Buildfile erzeugen, welches bereits Abhängigkeiten zu weiteren konfigurierten Libraries vermerkt. Dieses Buildfile bildet die Basis für das Project Buildfile.

Wir erstellen für jedes Projekt ein Buildfile, welches wir jedoch nicht aus JDeveloper heraus aufrufen. Grundsätzlich könnte der Aufruf auch durch die bereits vorhandenen Ant Integration in JDeveloper vorgenommen werden, allerdings ist der im folgenden beschriebene Ansatz flexibler.
Zu diesem Zweck gibt es ein zentrales Projekt „Deployment Process“, welches einen Wrapper über alle Projekte darstellt.

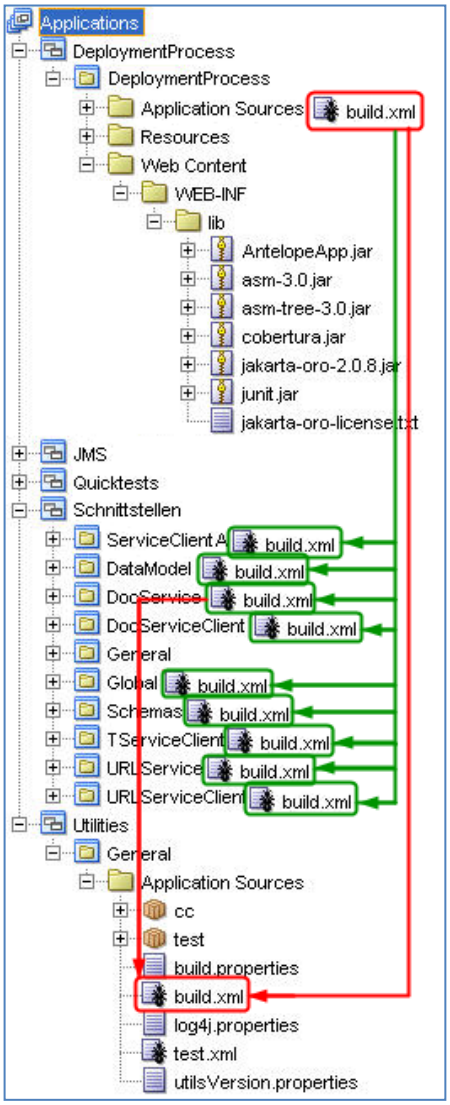


Abb. 9 Apache Ant - Globale Struktur der Projekte

In diesem Projekt wird gesteuert, welche Projekte in ein Deployment Paket gepackt werden. Es ist möglich, Projekte einzeln oder in der richtigen Reihenfolge bzw. Abhängigkeit zu erstellen. Weiterhin ist eine zentrale Vergabe von Versionsnummern für jedes einzelne Projekt möglich. Diese Versionsnummern werden z.B. in die Anwendungen (EAR Files), Bibliotheken (JAR Files) und in Datenbank SQL Skripten eingetragen. Die SQL Skripte werden verwendet, um Versionsinformationen zentral in der Datenbank sowie im Bugtracking System zu speichern.

Um die Übersichtlichkeit zu erhöhen wurde mit Tigris Antelope ein GUI für das zentrale Buildfile eingesetzt. Der Einsatz ist unkritisch und kann in bestehenden Umgebungen mit großen bis sehr großen Buildfiles besonders Sinn machen. Für eine sinnvolle Darstellung der Targets ist das Buildfile unter Umständen an die Namenskonventionen von Antelope anzupassen, die jedoch durchaus gebräuchlich sind.

So werden z.B. Sub-Targets mit einem Bindestrich begonnen und können dann durch entsprechende Konfiguration von der Anzeige der verfügbaren Targets ausgeschlossen werden.

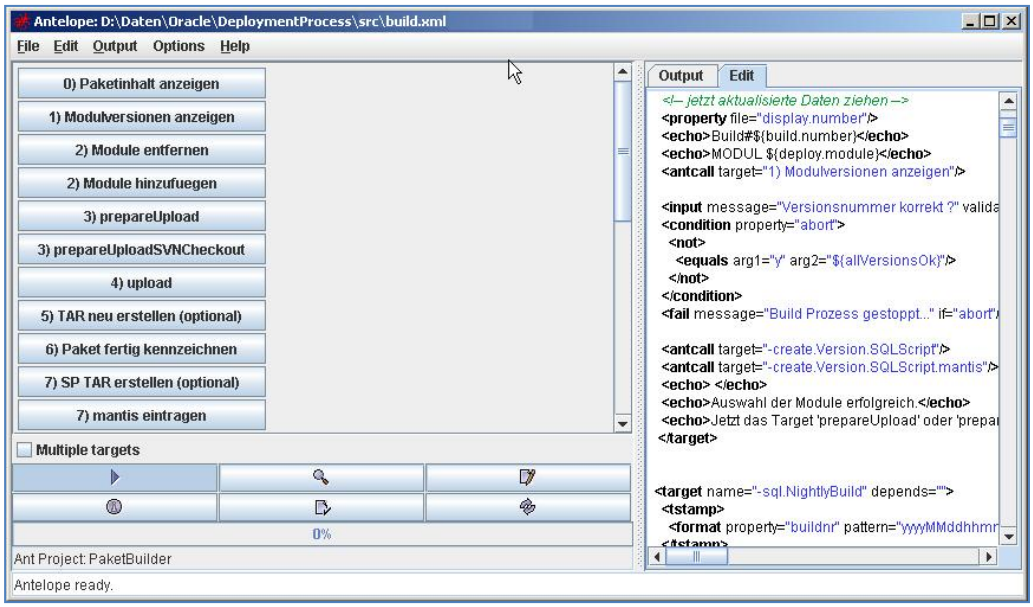


Abb. 10 Tigris Antelope GUI für Apache Ant

Kontaktadressen:

Christian Feicke
Computacenter AG & Co. oHG
Services & Solutions
Mariendorfer Damm 1-3
D-12099 Berlin

Telefon: +49 172 34 32 556
E-Mail christian.feicke@computacenter.com
Internet: www.computacenter.de

Florian Schulze
Computacenter AG & Co. oHG
Services & Solutions
Mariendorfer Damm 1-3
D-12099 Berlin

Telefon: +49 172 831 13 14
E-Mail florian.schulze@computacenter.com
Internet: www.computacenter.de