ApacheCon Europe 05 18.–22. Juli 2005 Alle Infos im Heft! (ab S.43)

XML magazin & WEB SERVICES

XML magazin & WEB SERVICES

Deutschland € 9,80
Österreich € 10,20
Schweiz sFr 19,20

S&S

Technology | Integration | Business

3.05

XML Content Management

Effizient Inhalte mit XML verwalten

XML und MS Office

Die XML-Funktionen in Word

XSLT und .NET

XSL-Transformationen leicht gemacht

PDF-Generierung

Die wichtigsten XSL-FO-Elemente

SVG mobil

Anwendungen mit SVG Tiny und SVG Basic

4 195893 609808 03

Auf den Zahn gefühlt

SOAPtest 3.0 angewandt

Serviceorientierte Architekturen, die meist auf dem Web-Service-Standard SOAP basieren, verlangen ein Höchstmaß an Verfügbarkeit und Verlässlichkeit. Um dies zu gewährleisten, sind umfangreiche Tests, nicht nur die ohnehin obligatorischen Unit-Tests, sondern auch Konformitätstests und Lasttests, erforderlich. Um diesen teils immensen Aufwand zu bewältigen, empfiehlt sich der Einsatz von Test-Werkzeugen, die zumindest einen Teil der Arbeit automatisieren können. Eine dieser Anwendungen ist SOAPtest aus dem Hause Parasoft.

von Martin Szugat

Seit der ersten Version von SOAPtest hat sich im Umfeld der Web-Services-Technologie einiges getan: Es kamen zahlreiche, ergänzende Standards hinzu und auch SOAP hat zwei kleinere Versionswechsel erfahren. Diese Änderungen spiegeln sich auch in SOAPtest wider: So werden in der aktuellen Version neben SOAP 1.2, WSDL 1.0 und XML Schema 1.0 folgende Standards unterstützt:

- Web Service Interopability (WS-I): Basic-Profile-1.0-Konformitätstests
- WS-Security, SAML, Username Token und X.509
- XML Encryption und XML Signature
- WS-Addressing

Die HOBITs

Der praktische Einsatz der Testsoftware wird im Folgenden anhand eines konkreten Projektes, der HOBIT-Initiative (**H**elmholtz **O**pen **B**io **I**nformatics **T**echnology), beschrieben [1]. Die HOBIT-Gruppe umfasst verschiedene Universitäten und Einrichtungen aus Deutschland und hat sich zum Ziel gesetzt, die Vernetzung von Bioinformatik-Anwendungen mittels Web Services voranzutreiben. Das hier vorgestellte und öffentlich zugängliche Projekt [2] trägt dazu bei, indem es einen Thesaurus für Proteinnamen und Datenbankidentifiern bietet. Mittels einer Microsoft-Office-Erweiterung namens ProTag wird diese Funktionalität in Form von Smart Tags in die bekannten Anwendungen Word, Excel und PowerPoint integriert. Dabei ist die Funktionalität für die Abbildung von Proteinnamen auf IDs und die für das Tagging von Proteinnamen und IDs in beliebigen Texten auf zwei separate Schnittstellendefinitionen verteilt, für die jeweils getrennte WSDL-Dokumente zur Verfügung stehen: *BiologicalNameService.wsdl* und *BiologicalMarkupService.wsdl*. Des Weiteren existieren verschiedene Implementierungen dieser Schnittstellen: ProThesaurus implementiert sowohl einen Biological Name Service als auch einen Biological Markup Service, wobei letzterer nur Identifier erkennt. Für das Tagging von Proteinnamen steht hingegen LiMB zur Verfügung. Um die Funktionsfähigkeit und Verfügbarkeit dieser Web Services sicherzustellen, ist eine Reihe von Tests erforderlich, die SOAPtest allesamt unterstützt:

- Konformitätstests: Inwieweit sind die Web Services kompatibel zu verschiedenen Clientarchitekturen (Beispiel .NET, Java oder Perl)? Und: Erfüllen sie den WS-I-Standard?
- Unit-Tests: Wie verhalten sich die Web Services bei verschiedenen Anfragen, insbesondere bei ungültigen Eingaben?
- Funktionstests: Ist die Datenhaltung und -bereitstellung konsistent, sprich ist die Abbildung von IDs auf Proteinnamen und wieder zurück eindeutig?
- Regressionstests: Welche Unterschiede, die eventuell die Clients negativ beeinflussen, ergeben sich im Laufe der verschiedenen Web-Services-Versionen?
- Lasttests: Welchen Ansturm an Anfragen können die Web Services bewältigen?
- Clienttests: Halten sich die Clients an die im WSDL-Dokument hinterlegten (Kommunikations-)Regeln?

Die unterschiedlichen Tests werden dabei je nach Aufgabe in verschiedenen Umgebungen ausgeführt. So existiert neben der Produktionsumgebung, die öffentlich unter [2] verfügbar ist, eine nahezu identische Testumgebung. Erst wenn ein Web Service oder eine neue Version eines Web Services alle Tests erfolgreich bestanden hat, wird er auf dem Produktionsserver eingespielt. Dort findet ein erneuter Testdurchlauf statt, um die korrekte Installation des Web Services zu überprüfen. Die ersten Tests finden jedoch in der Entwicklungsumgebung, sprich auf den lokalen Rechnern der einzelnen Entwickler statt, und dienen der Validierung von funktionalen Änderungen und Erweiterungen.

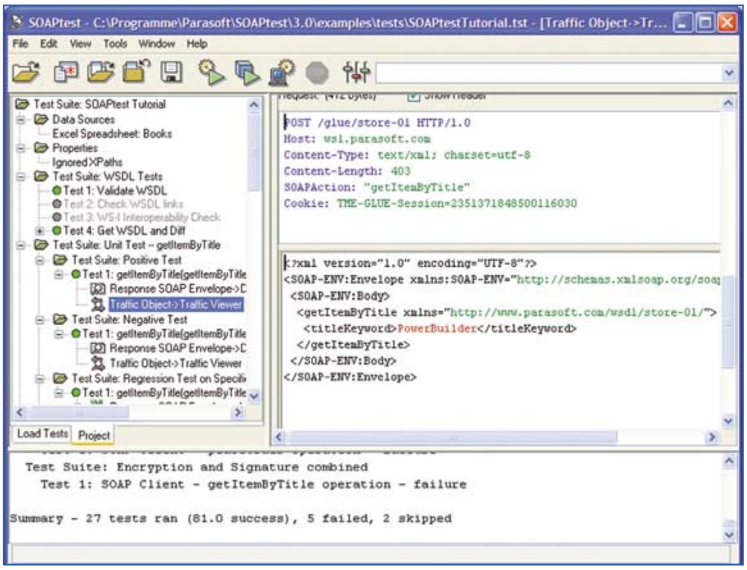
Schlüssel zum Erfolg

SOAPtest kann von [3] für verschiedene Plattformen, darunter Windows und Unix, bezogen werden. Hierfür ist eine Registrierung erforderlich, in deren Folge dem Benutzer ein Download-Key per E-Mail zugesandt wird. Dieser Schlüssel ist für einen Tag gültig. Nach dem erfolgreichen Download und der Installation wird beim ersten Start von SOAPtest ein Passwort verlangt. Tatsächlich handelt es sich hierbei um einen Lizenzschlüssel, welcher in einer zeitlich begrenzten Form von Parasoft für die Evaluierung von SOAPtest zur Verfügung gestellt wird. Hierbei ist eine Machine ID zu übermitteln, die bei der Passwort-Eingabe eingeblendet wird. Für Mehrplatzinstallationen wird zudem ein License Server angeboten, über den die einzelnen SOAPtest-Installationen ihre Lizenzschlüssel beziehen können.

In Verbindung mit dem ebenfalls bei Parasoft erhältlichen Group Reporting System (GRS) können alle Testergebnisse zudem zentral gespeichert und über eine konfigurierbare Weboberfläche von den beteiligten Entwicklern und Projektmanagern eingesehen und analysiert werden.

Konformismus

Im Mittelpunkt der Aktivitäten von SOAPtest stehen die WSDL-Dokumente, welche nicht nur die Schnittstellen der Web Services beschreiben, sondern auch deren Endpunkte, also die konkreten URLs, unter denen ein Web Service verfügbar ist. Über den Menüpunkt FILE | NEW wird ein neues Testprojekt angelegt und mit einem WSDL-Dokument assoziiert. Der SOAP Test Wizard legt hierbei für die vorhandenen Web-Services-Methoden automatisch entsprechende Unit-Tests an. Sofern mehr als ein Endpunkt definiert wurde, werden für die einzelnen Methoden jeweils mehrere Tests angelegt. Für den ProThesaurus und den LiMB Web Service wird auf der Heft-CD ein vorkonfiguriertes Testprojekt zur Verfügung gestellt.



```
POST /glue/store-01 HTTP/1.0
Host: wsl.parasoft.com
Content-Type: text/xml; charset=utf-8
Content-Length: 403
SOAPAction: "getItemByTitle"
Cookie: TME-GLUE-Session=2351371848500116030
```

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soa
 <SOAP-ENV:Body>
  <getItemByTitle xmlns="http://www.parasoft.com/wsdl/store-01/">
   <titleKeyword>PowerBuilder</titleKeyword>
  </getItemByTitle>
 </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
 Test Suite: Encryption and Signature combined
   Test 1: SOAP Client - getItemByTitle operation - failure

Summary - 27 tests ran (81.0 success), 5 failed, 2 skipped
```

Abb. 1: Die Benutzeroberfläche von SOAPtest

Durch die Aktivierung der Option *Create WSDL Tests* im FILE | NEW-Dialog wird zusätzlich zu den Unit-Tests eine weitere Test Suite (namens *WSDL Tests*) erzeugt, welche das WSDL-Dokument selbst prüft. So wird unter anderem die Verfügbarkeit der dort beschriebenen Endpunkte ebenso überprüft wie die korrekte WSDL- und XML-Schema-Syntax. Der weitaus interessanteste und hilfreichste Test ist jedoch der, der die Einhaltung des WS-I-Standards [4] kontrolliert. So ergab sich beispielsweise beim Testen der Biological Name und Mark-up Services, dass die mit Apache Axis [5] erzeugten Web-Service-Beschreibungen nicht konform zu diesem Standard sind.

Happy Tree Friends

Falls wie im hier vorgestellten Projekt mehrere WSDL-Dokumente vorliegen, so können diese auch in einem gemeinsamen Testprojekt zusammengefasst werden, indem das Projekt in verschiedene Testsuiten unterteilt wird. Diese lassen sich wiederum in weitere Testsuiten untergliedern, die letztlich die einzelnen Tests aufnehmen. Selbst externe Testdateien lassen sich als Referenz oder als Kopie in eine Testsuite aufnehmen. Diese hierarchische Anordnung spiegelt sich auch in der Benutzeroberfläche von SOAPtest (Abbildung 1) wider: In einer Baumstruktur können Tests und Testsuiten erstellt, verschoben und wieder gelöscht werden. In der rechten Ansicht wird der jeweils in der Baumstruktur selektierte Test beziehungsweise die ausgewählte Testsuite konfiguriert. Über die am oberen Rand befindlichen Schaltflächen können einzelne Tests oder ganze Testsuiten zur Ausführung gebracht werden. Der Fortgang des Testlaufes wird dabei protokolliert. Die Ergebnisse des Tests werden in SOAPtest angezeigt, können aber zusätzlich auch an beliebig viele Tools weitergegeben oder in eine Datei geschrieben werden. So kann die Anzeige beispielsweise im Browser erfolgen, was im Falle der WS-I-Tests zum Beispiel die Generierung eines detaillierten WS-I Profile Conformance Reports zur Folge hat.

Regression statt Progression

Unit-Tests werden beim Anlegen eines neuen Testprojektes auf Wunsch automatisch vom SOAP Test Wizard erzeugt. Allerdings weisen die generierten Tests mehrere Mankos auf: Erstens verwenden sie Zufallswerte für die Parametereingabe und zweitens werden die Ausgabewerte nicht validiert. Es wird folglich nur festgestellt, ob eine Web-Service-Methode überhaupt ein Ergebnis liefert oder ob sie eine Ausnahme auslöst. Sowohl das Ergebnis als auch die Anfrage selbst werden dabei in einem Traffic Viewer festgehalten.

Das erste Problem ist schnell behoben: Ein Klick auf den entsprechenden Baumeintrag offenbart die Einstellungen für den Test. Unter PARAMETERS wird anstelle von *Automatic* die Option *Form* gewählt, und die Parameter werden mit den entsprechenden Werten belegt. Alternativ kann die Option *Method* gesetzt werden, und die Eingabewerte werden von einem Java-Programm oder einem JavaScript- beziehungsweise Python-Skript erzeugt. Indem als Ansichtsmodus Literal XML gewählt wird, kann selbst die SOAP-Nachricht in XML editiert werden. Mit Form XML steht zudem eine hierarchische Datensicht zur Verfügung. Und Scripted XML bietet die Möglichkeit der XML-Generierung mittels der bereits genannten Programmiersprachen.

Das zweite Problem verlangt nur unwesentlich mehr vom Benutzer: Ein Klick mit der rechten Maustaste auf den jeweiligen Test lässt ein Kontextmenü erscheinen. Über den Menüpunkt Add Output | Response | SOAP Envelope | New Output stehen verschiedene Werkzeuge für die Analyse der SOAP-Antwortnachricht zur Verfügung. Um einen Unit-Test durchzuführen, eignet sich insbesondere das Modul *Diff*. Es vergleicht die Ausgabe des Web Service mit zuvor festgesetzten Werten, wobei verschiedene Modi zur Verfügung stehen: Binary, Text und XML. Bei letzterem stehen außerdem die bekannten Ansichten Literal XML und Form XML zur Verfügung. Der XML-Modus bietet zudem den Vorteil, dass bestimmte Teile der SOAP-Nachricht vom Vergleich ausgeschlossen werden können. Sie werden mithilfe von XPath definiert.

Das *Diff*-Modul erzeugt seinerseits eine Ausgabe, falls die tatsächlichen Werte nicht mit den erwarteten übereinstimmen. Diese Ausgabe lässt sich wiederum an ein anderes Modul weiterleiten, um beispielsweise Fehler in einer Datei zu protokollieren. Bei Regressionstests findet das *Diff*-Modul erneut Anwendung. Allerdings nimmt SOAPtest hier dem Entwickler die Arbeit ab, indem es den Befehl *Create Regression Control* im Kontextmenü eines Tests bietet. Hierbei wird die jeweilige Web-Service-Methode einmalig aufgerufen und das *Diff*-Modul wird mit der zurückgegebenen SOAP-Nachricht initialisiert. Bei den folgenden Tests werden demnach die aktuellen Rückgabewerte mit dem gespeicherten Ergebnis vergleichen. Abweichungen werden als Fehler gemeldet. Um einen solchen Test auf den neuesten Stand zu bringen, genügt die Ausführung des Befehls *Update Regression Control*.

Bürger aus Konsolien

Anhänger der Konsole, wie zum Beispiel die Administratoren der Produktionsumgebung, die sicherstellen müssen, dass ein Web Service korrekt eingespielt wurde, werden bei SOAPtest ebenfalls berücksichtigt. Hierzu gibt es das Kommandozeilenwerkzeug *st.exe* für Windows und *soaptest* für Unix. Bei ersterem muss zusätzlich der Schalter *-cmd* gesetzt werden, um die Anzeige der Benutzeroberfläche zu unterdrücken. Mithilfe des Schalters *-runtest* werden die in einer Testdatei definierten Tests zur Ausführung gebracht. Als letztes folgt der Pfad der Testdatei:

Hierarchische Datensicht mit Form XML

```
st.exe –cmd –runtest C:\Test\Services.tst
```

Bei der für diesen Artikel verwendeten Version (3.0.2) für Windows trat das Problem auf, dass das Programm *st.exe* nur aus dem Installationsverzeichnis von SOAPtest aufgerufen werden konnte. Wurde *st.exe* über eine absolute Pfadangabe gestartet, so wurde eine *NullPointerException* geworfen. Anstelle der Testdatei kann ebenso ein Kommandoskript übergeben werden, welches den Vorteil bietet, dass mehrere Testdateien sowie zusätzliche Optionen spezifiziert werden können. In diesem Fall ist als Schalter *-run* zu setzen:

```
st.exe –cmd –run C:\Test\Services.scr
```

Bei den Kommandoskriptdateien handelt es sich um einfache Textdateien, die mit beliebig vielen Einträgen wie beispielsweise dem folgenden gefüllt werden können:

```
runTest C:\Test\Services.tst -router http://localhost:8080/
http://services.bio.ifi.lmu.de:1046/
```

Bei diesem Beispiel werden mithilfe des Arguments *-router* die lokalen URLs der Web-Service-Endpunkte temporär durch entfernte URLs ersetzt. Dies ist beispielsweise dann hilfreich, wenn die Tests für die Testumgebung ausgelegt wurden, nun jedoch in der Produktionsumgebung geprüft werden sollen. Allerdings sind zuvor geringfügige Änderungen an der Testdatei durchzuführen, denn standardmäßig ist in den Testeinstellungen die Option *Constrain to WSDL* aktiviert, sodass die Endpunkte an die im WSDL-Dokument hinterlegten URLs gebunden sind und sich infolgedessen nicht ändern lassen. Nachdem diese Option deaktiviert wurde, können die URLs manuell bearbeitet oder mittels *-router* ersetzt werden.

Die übrigen Parameter beeinflussen die Ausgabe des Fehlerberichts. So ist als Ausgabeformat Text, XML und HTML möglich und der Bericht lässt sich auf der Konsole ausgeben, in eine Datei schreiben oder an eine E-Mail-Adresse verschicken. Für das Ausführen von Lasttests steht eine gesonderte Anweisung zur Verfügung: *loadtest*. Anstatt eines einfachen Kommandoskripts können ebenso Skriptdateien in den Sprachen JavaScript oder Python übergeben werden. Für diese Sprachen stellt SOAPtest ein eigenes Scripting-API bereit.

Die Clients, die Last und der ganze Rest

Funktionale Tests, die auch als Szenariotests bezeichnet werden, gehen weit über die Möglichkeiten einfacher Unit-Tests hinaus, indem sie einzelne Tests miteinander verknüpfen. Dies erfolgt mithilfe des *XML Data Bank*-Moduls, welches die Ausgabe ein oder mehrerer Tests speichert und für die Eingabe an weitere Tests bereitstellt. So lassen sich komplexe Abläufe aufeinander folgender Web-Service-Aufrufe simulieren und auf ihre korrekte Funktion hin überprüfen.

Besonders nützlich für das Testen der Clients, aber auch für das schnelle Prototyping, ist die Fähigkeit von SOAPtest, einen Webserver zu emulieren. Hierfür wird SOAPtest mit einem vorkonfigurierten Tomcat-Server inklusive des Apache-Axis-Moduls ausgeliefert. Über die grafische Benutzeroberfläche von SOAPtest kann nicht nur der Webserver gestartet und gestoppt werden, es können auch neue Web Services registriert werden, indem entweder WSDL-Dokumente oder aber WSDD-Dokumente (**W**eb **S**ervice **D**eployment **D**escriptor) importiert werden. Dabei werden automatisch die passenden Stub-Klassen generiert, die anschließend in einem rudimentären Java-Editor angepasst und schließlich kompiliert werden können. Dank eines SOAPtest-API lassen sich selbst Proxies auf diese Weise realisieren.

Am Ende einer jeden Testreihe stehen die Lasttests, auch Stresstests genannt. Hierbei wird der Web Service einer bestimmten Anzahl von virtuellen Benutzern ausgesetzt und dabei das Verhalten des Web Services beobachtet. Die maximale Anzahl an möglichen Benutzern hängt von der jeweils erworbenen Lizenz ab. Die Kosten für eine Lizenz mit 100 virtuellen Benutzern betragen stolze 17.900 €. Der Preis für Versionen ohne Unterstützung von Lasttests beginnt bei 4.500 €.

Benutzer können auch auf mehrere Rechner verteilt werden, wofür ein eigener Load Test Server bereitsteht. Das Verhalten der Benutzer wird mithilfe so genannter Szenarios bestimmt. Ein einfaches Szenario nennt sich *Linear Increase* und simuliert eine linear ansteigende Last. Eine Alternative hierzu ist das *Bell*-Szenario, bei dem die Verteilung einer parametrisierbaren gaußschen Glockenkurve folgt. Protokolliert wird hierbei stets die aktuelle Ausführungszeit (durchschnittlich, minimal und maximal), die Anzahl bestandener und fehlgeschlagener Tests sowie die Benutzerzahl. Die Anzeige erfolgt dabei als Graph.

Fazit

SOAPtest bietet in seiner dritten Version viele nützliche und wichtige Funktionen für das Testen von Web Services: Konformitäts-, Unit-, Funktions-, Regressions-, Last- und Clienttests. Um diese sinnvoll zu nutzen, bedarf es allerdings eines gewissen Einarbeitungsaufwands. Denn obwohl die Benutzeroberfläche in einem schlichten Gewand auftritt, verbergen sich dahinter viele nützliche Funktionen, die meist erst durch das Lesen der Dokumentation entdeckt werden. So können beispielsweise die Testwerte auch von einer Datenbank oder gar aus einer Excel- oder CSV-Datei bezogen werden. Alles in allem ist SOAPtest eine unverzichtbare Hilfe bei der Entwicklung von zuverlässigen Web Services. ●

Martin Szugat ist seit vielen Jahren als freischaffender Fachautor im Bereich der Softwareentwicklung tätig. Zu seinen Schwerpunkten zählen XML und .NET. Bei Fragen erreichen Sie ihn unter Martin.Szugat@GMX.net. Über aktuelle Neuigkeiten rund um XML berichtet er auch regelmäßig in seinem Blog unter www.aboutxml.de.

Links & Literatur

[1] HOBIT: hobit.sourceforge.net

[2] ProThesaurus: services.bio.ifi.lmu.de/prothesaurus/

[3] Parasoft: www.parasoft.com

[4] WS-I: www.ws-i.org

[5] Apache Axis: ws.apache.org/axis/