

JavaTMmagazin

Internet & Enterprise Technology

Tomcat 4

Alles über die neue Generation von
Java Servlets und JavaServer Pages

Kryptographie in Java & XML

Notwendig für eBusiness: Sicherer
Datenaustausch

Extreme Programming

Radikale Methoden für Java-Entwickler

Enterprise

Dynamische Business-Objekte entwerfen

Web Services

SOAP-Anwendungen mit Glue realisieren

Entwicklung

Neue Features in Visual Age for Java 4.0

XML-Testframework

Effizienz durch Unit-Tests mit JXUnit



Sonderdruck

Java Magazin, Ausgabe 12.2001

Im Blickpunkt: VisualAge for Java 4.0 und sein Nachfolgeprodukt

von Stefan Schäffer und Walter Schilder

New Age

Mit VisualAge for Java 4 bietet IBM seit August dieses Jahres die neueste und gleichzeitig letzte Version des erfolgreichen Java-Entwicklungswerkzeuges an. In diesem Artikel stellen wir die neuen Features dieser Version vor. Ferner liefern wir einen Ausblick, wie die zukünftige Java-Entwicklungsumgebung der IBM aussehen wird.



VisualAge for Java 4.0 – Neue Features

In Bezug auf neue Funktionalität gibt es bei VisualAge for Java 4.0 nur relativ wenig zu berichten. Die neue Version des Java-Entwicklungswerkzeuges bietet im Vergleich zur Vorgängerversion 3.5.3 nämlich kaum Neuerungen. Positiv fällt zunächst jedoch auf, dass sich die Stabilität des Werkzeugs noch einmal verbessert hat. Die IBM hat in die neue Version offensichtlich einige Bugfixes eingearbeitet, um die bereits relativ ro-

buste Vorgängerversion noch stabiler zu machen. Der seit Version 3.5.3 auftretende Fehler bei der Quelltext-Formatierung wurde aber leider nicht behoben. Damit ergibt sich nach wie vor der Effekt, dass bei Verwendung der Formatierungseinstellung „Opening braces begin new line“ die öffnende geschweifte Klammer von auf Schlüsselwörter (if, while, for, try, etc.) folgenden Anweisungsblöcken nicht korrekt formatiert wird.

In Version 4.0 neu hinzugekommen ist die Möglichkeit, Enterprise JavaBeans nach dem EJB-Standard 1.1 zu exportieren. Das bedeutet, dass bei Verwendung der neuen Export-Funktion nun ein Deployment Deskriptor im XML-Format erzeugt wird und nicht wie bisher die im Entwicklungswerkzeug vorgenommenen Deployment-Einstellungen in Form einer serialisierten Klasse abgelegt werden. Die neue Export-Funktion wurde

aufgrund der Kompatibilität zum WebSphere Application Server 4.0 hinzugefügt und äußert sich lediglich durch einen zusätzlichen Eintrag („Export EJB 1.1 Jar...“) im Menüpunkt EJB | EXPORT der EJB-Ansicht (siehe Abb. 1). Dabei ist jedoch zu beachten, dass die erweiterte Export-Funktion nach der Installation des Programms nicht sofort zur Verfügung steht – die entsprechende Zusatzfunktion namens „Export Tool for Enterprise JavaBeans“ muss zuerst über den Menüpunkt FILE | QUICK START | FEATURES | ADD FEATURE installiert werden.

Überarbeitet wurden in der neuen Version auch die Konnektoren, mit denen eine relativ einfache Anbindung von Java-Anwendungen an verschiedene andere Systeme ermöglicht wird. Folgende Systeme können derzeit mit Hilfe der mitgelieferten Konnektoren angebunden werden:

- IBM CICS Transaction Server (konform zur J2EE Connector Architecture)
- IBM Encina
- IBM IMS
- IBM MQSeries
- IBM Host on-Demand
- JD Edwards OneWorld (konform zur J2EE Connector Architecture)
- Oracle Applications (konform zur J2EE Connector Architecture)
- Peoplesoft (konform zur J2EE Connector Architecture)
- SAP Release 3 (konform zur J2EE Connector Architecture)

Wie aus obiger Aufzählung zu entnehmen ist, ist gut die Hälfte der in Version 4.0 mitgelieferten Konnektoren nun konform zur Java 2 Enterprise Edition Connector Architecture, einem von Sun Microsystems veröffentlichten Standard, der die Schnittstellen zwischen Application Server und Back-End-Systemen genauer festlegt. Ziel dieser Spezifikation ist es, die Anbindung von Back-End-Systemen an die J2EE-Plattform weitestgehend zu standardisieren. Dass diejenigen im Rahmen von VisualAge for Java mitgelieferten Konnektoren, die zur J2EE Connector Architecture konform sind, lediglich als Beta-Version vorliegen, liegt ausschließlich daran, dass sich die

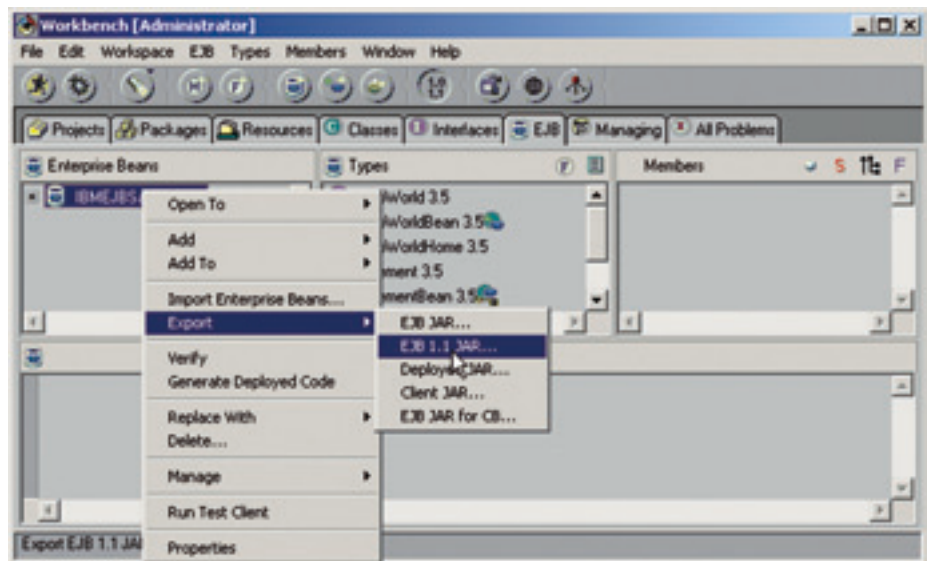


Abb. 1: Die neue Funktion Export EJB 1.1 Jar... erzeugt einen Deployment Deskriptor in XML

von Sun Microsystems definierte Spezifikation lange Zeit im Stadium Proposed Final Draft (einer Art Beta-Version des Standards) befand. Seit August liegt die knapp 200 Seiten umfassende J2EE Connector Architecture aber als Final Release (Version 1.0) vor.

Stärken und Schwächen

Da mit der neu erschienenen Version wenig neue Funktionalität hinzugekommen ist, bleiben die Vor- und Nachteile dieses Entwicklungswerkzeuges weitestgehend erhalten.

Als großer Pluspunkt von VisualAge for Java ist sicherlich die integrierte WebSphere Testumgebung (engl. WebSphere Test Environment, WTE) zu sehen. Damit ist es vor allen Dingen möglich, Enterprise JavaBeans ohne vorherigen Export in einen separaten Application Server ausgiebig auf Funktionalität zu testen. Ähnliches gilt für Servlets und JSPs; auch sie lassen sich sofort in der Entwicklungsumgebung ausführen. Dabei geht VisualAge for Java aber noch einen Schritt weiter: Es stellt nicht nur eine Ausführungsumgebung zur Verfügung, sondern erlaubt auch das Debuggen sämtlicher erstellter Klassen, egal ob sie Bestandteil von EJBs, JSPs, Servlets oder auch Stand-alone-Anwendungen sind. Durch den eingebauten inkrementellen Compiler lässt sich der Quelltext bereits im laufenden Betrieb

modifizieren und erneut testen, ohne dass ein Neustart der Anwendung notwendig ist. Dies ist insbesondere bei der Entwicklung von serverseitigen Anwendungen von großem Vorteil, da hier das Starten der Ablaufumgebung sehr zeitintensiv sein kann. Die durch den Einsatz der WTE bedingte Zeitersparnis ist besonders bei der Entwicklung von EJBs erheblich, da ein Debugging im Application Server in der Regel sehr mühsam ist.

VisualAge for Java bietet mit dem sogenannten Visual Composition Editor (VCE) darüber hinaus auch umfangreiche Unterstützung für die auf dem JavaBeans-Konzept basierende grafische Programmierung. Der VCE ist ein mächtiges Werkzeug, welches weitaus mehr Funktionalität bietet als der derzeit weit verbreitete GUI-Builder. Richtig angewendet lassen sich mit dem VCE komplette Anwendungen weitestgehend grafisch erstellen, eine Möglichkeit, die bereits in zahlreichen Projekten geschätzt und verwendet wird.

Als zusätzlicher Vorteil erweist sich in der Praxis auch, dass VisualAge eine umfangreiche Menge an Code-Generatoren zur Verfügung stellt, angefangen bei der Erstellung von Attributen, Methoden oder Klassen bis hin zur Entwicklung von grafischen Benutzungsoberflächen, Servlets oder EJBs. Der Entwickler wird durch eine Reihe von Wizards unterstützt. Auch



Abb. 2: Funktionalität des WebSphere Studio Application Developers

die Anbindung an andere Programmiersprachen, wie beispielsweise C++, oder die Integration von Back-End-Systemen (siehe Konnektoren) läuft durch die Verwendung von entsprechenden Wizards weitestgehend automatisiert ab.

Trotz dieser Vorteile hat die Entwicklungsumgebung auch einige Schwächen, die den Einsatz in der Praxis manchmal etwas schwierig gestalten. Als Erstes ist hier zu kritisieren, dass von VisualAge 4.0 immer noch das bereits veraltete JDK 1.2.2 unterstützt wird, obwohl von Sun Microsystems bereits neuere Versionen des Java Development Kits (JDK) zur Verfügung stehen. Auch beim EJB-Standard sieht es nicht besonders gut aus. Wer mit VisualAge entwickelt, erstellt Enterprise JavaBeans nach dem EJB-Standard 1.0. Das Entwicklungswerkzeug bietet zwar zahlreiche Erweiterungen, die erst durch neuere Spezifikationen abgedeckt werden, wie beispielsweise Entity Beans oder die Möglichkeit, Assoziationen zwischen EJBs zu erstellen. Trotzdem ist der EJB-Standard 1.1 noch nicht vollständig implementiert. Ein Schritt in die richtige Richtung ist dabei die neu hinzugekommene Exportfunktion.

Dabei stellt sich sicherlich die Frage, warum die IBM den aktuellen Standards so weit hinterherhinkt, obwohl sie doch an vielen Spezifikationen so maßgeblich

beteiligt ist. Dies liegt im Wesentlichen an dem VisualAge zugrunde liegenden Software Configuration Management (SCM) Werkzeug namens ENVY. Aus vorangegangenen Entwicklungswerkzeugen wie VisualAge for Smalltalk übernommen, entwickelte sich ENVY von einem einstigen Pluspunkt zu einem wesentlichen Nachteil. ENVY ist nämlich im Gegensatz zu heute üblichen SCM-Werkzeugen nicht dateibasiert, sondern erlaubt eine Versionierung auf unterschiedlichen Ebenen. Damit ergibt sich unter Umständen der Vorteil einer sehr viel feineren Granularität der Versionierung. Dem steht jedoch die Tatsache gegenüber, dass ENVY lediglich die Ablage von Java-Quelltexten im Repository ermöglicht. Zusätzliche Ressourcen müssen dagegen anderweitig verwaltet werden. Dies führt in der Praxis häufig zur Verwendung von weiteren SCM-Werkzeugen, wie beispielsweise CVS, und den damit verbundenen Nachteilen (keine vollständige Integration in VisualAge for Java, doppelte Datenhaltung, etc.). Zusätzliche Nachteile ergeben sich durch die sehr enge Verknüpfung mit der verwendeten Ablaufumgebung. Aus diesem Grund gestalten sich Änderungen am zugrunde liegenden JDK als relativ komplex. Bei jedem Wechsel des JDK ist somit ein erheblicher Entwicklungsaufwand notwendig.

Wegen diesen Nachteilen hat man bei IBM die tief greifende Entscheidung getroffen, sich bei zukünftigen Entwicklungen von ENVY zu lösen. Da ENVY aber, wie bereits erwähnt, ein integraler Bestandteil von VisualAge for Java ist, geht dies im Wesentlichen mit einer kompletten Neuentwicklung des Werkzeugs einher. Aufgrund der allgemeinen Strategieausrichtung der IBM wird dieses Entwicklungswerkzeug auch nicht mehr wie gewohnt den Namen VisualAge for Java tragen, sondern unter dem Namen WebSphere Studio Application Developer (WSAD) vermarktet werden. Mit der Version 4 enden also sämtliche Weiterentwicklungen an VisualAge for Java; es beginnt die neue WSAD-Ära. Derzeit besteht von Seiten IBMs aber kein Druck, sofort auf das neue, bisher noch nicht vollständig ausgereifte Werkzeug umzusteigen, denn es wurde zugesichert, noch bis Juni 2003 Support für VisualAge for Java 4.0 zu gewährleisten. Dies beinhaltet auch die Bereitstellung von Fixpacks, falls in Zukunft schwerwiegende Fehler bekannt werden sollten.

WebSphere Studio Application Developer – Was ist das?

Wie bereits angedeutet, ist das Ergebnis der Neuentwicklungen schon heute sichtbar, da das als Beta-Version verfügbare Werkzeug WSAD kostenlos aus dem Internet heruntergeladen werden kann. Darüber hinaus wird es derzeit auch als kostenlose Beilage zu VisualAge for Java 4.0 ausgeliefert. Die beigelegte Version des WSAD ist für eine Beta-Version schon bemerkenswert stabil. Dies liegt wohl hauptsächlich in der Tatsache begründet, dass an diesem Entwicklungswerkzeug bereits seit über zwei Jahren entwickelt wird.

WSAD bietet eine umfangreiche Unterstützung für zahlreiche Tätigkeiten, die einen Softwareentwicklungszyklus ausmachen. Die von WSAD bereitgestellte Funktionalität lässt sich dabei in zwei Bereiche unterteilen (siehe Abb. 2).

Zunächst erläutern wir die Funktionen, welche auch in das neu entwickelte Produkt WebSphere Studio Site Developer (WSSD) eingegangen sind. Beim WSSD handelt es sich um den Nachfol-

ger von IBM WebSphere Studio, einem umfangreichen Werkzeug zur Entwicklung von Web-Auftritten. WebSphere Studio beschränkt sich im Gegensatz zu anderen Werkzeugen jedoch keineswegs auf die Gestaltung von statischen HTML-Seiten, sondern bietet vielmehr als Kernfunktionalität eine umfangreiche Unterstützung für die Erstellung von dynamischen Web-Seiten in Form von JavaServer Pages (JSPs). WSSD beinhaltet die folgende Teilfunktionalität des WSAD:

- Java-Entwicklungsumgebung (Editor, Suchwerkzeuge, inkrementeller Compiler, Debugger etc.)
 - Unterstützung für die Entwicklung im Team (Anbindung an SCM-Werkzeuge, wie beispielsweise CVS)
 - Auf den jeweiligen Entwickler (Web-Entwickler, Java-Entwickler) zugeschnittene Ansichten für ein Projekt (vordefinierte Ansichten sind durch den Anwender anpassbar)
 - Wizards zur Erstellung von Web Services (Einbindung von bereits bestehenden Web Services möglich)
 - XML-Werkzeuge (Erstellung von DTDs und Schemas, XLST-Unterstützung etc.)
 - Unterstützung von JSPs (Wizards zur Einbindung von JavaBeans und Scriptlets, Unterstützung für Server Side Include, integrierte Testumgebung, komfortables Erstellen von statischen Inhalten, Einbindung von Skripten und Applets, uvm.)
- Darüber hinaus bietet der WebSphere Studio Application Developer noch weitere Funktionalität, welche im Folgenden aufgeführt ist:

- Wizards zur Realisierung von Datenbankverbindungen
- Werkzeuge zur Analyse und Optimierung des Laufzeitverhaltens (Analyse von Heap und Stack, CPU-Zeit etc.)
- Servlet-Entwicklung (Wizards zum Erstellen von Servlets, integrierte Testumgebung)
- EJB-Entwicklung (Wizards zum Erstellen von EJBs, Werkzeuge für das Mapping auf relationale Datenbanken, integrierte Testumgebung, etc.) und Deployment (Export-Funktionen, erstellen von EAR-Dateien)

Bei Verwendung des WSAD wird klar ersichtlich, dass die IBM hier sehr großen Wert auf eine umfangreiche Unterstützung bei der Entwicklung von EJBs gelegt hat. Positiv fällt neben den zahlreich vorhandenen Funktionen auf, dass der WSAD auch die volle EJB-Spezifikation 1.1 unterstützt. Die eingebaute WebSphere Testumgebung (WebSphere Application Server 4.01) bietet darüber hinaus schon während der Entwicklungsphase ein sehr gutes Laufzeitverhalten. Außerdem werden die beiden Werkzeuge WSSD und WSAD neben der Windows-Plattform (NT, 98, 2000 Millennium Edition) auch für Linux verfügbar sein.

Mit der Umstellung auf WSAD geht jedoch auch eine Änderung der von VisualAge for Java gewohnten Produkt-Editionen (Professional und Enterprise Edition) einher. Dies zieht für die WSAD Enterprise Edition einen gegenüber der VisualAge for Java Enterprise Edition höheren Preis nach sich. Deshalb empfiehlt die IBM ihren Interessenten, noch jetzt eine Lizenz für die VisualAge for Java Enterprise Edition zu erwerben. Im Zuge des meist damit verbundenen Upgrade-Rechts erhält der Kunde in diesem Fall dann kostenlos die Möglichkeit, mit der WSAD Enterprise Edition zu arbeiten.

Das Konzept

Neben der Ähnlichkeit in der Produktbezeichnung besitzen WSAD und WSSD aber noch weitere Gemeinsamkeiten. Zum einen sollen beide Entwicklungswerkzeuge noch im vierten Quartal 2001 als endgültige Versionen verfügbar sein. Obwohl die Beta-Version des WSAD einen Monat später als geplant erschienen ist, will die IBM diesen Zeitplan einhalten. Noch bemerkenswerter ist jedoch, dass beide Produkte auf Basis des gleichen technischen Konzepts realisiert wurden. Die dahinter liegenden Grundideen erscheinen dabei so interessant und vielversprechend, dass sie im Folgenden näher erläutert werden.

Die IBM hat für sämtliche auf Java basierenden Entwicklungsumgebungen ein gemeinsames Framework, die sogenannte WebSphere Studio Workbench (WSW) entwickelt. Hier wird bereits ein Großteil der gemeinsam benutzten Funktionalität



Abb. 3: Wizard zur Erstellung von EJBs

zur Verfügung gestellt. So beinhaltet die WSW neben einem komfortablen Text Editor mit Syntax Highlighting, Such- und Ersetzungsfunktionen etc. auch die Möglichkeit zur Anbindung an gängige SCM-Systeme. Darüber hinaus ist die WebSphere Studio Workbench so entwickelt, dass die verwendete Java-Laufzeitumgebung (JRE) für jedes Werkzeug bzw. jede Zielplattform unabhängig gewählt werden kann. Entwicklungswerkzeug und Laufzeitumgebung sind bei den auf der WSW basierenden Werkzeugen also nicht mehr so eng miteinander verknüpft, wie das zum Beispiel bei VisualAge for Java der Fall ist. Somit ist zu erwarten, dass neue JDKs auch in Zukunft viel einfacher und schneller integriert werden können. Basierend auf den von der WSW bereitgestellten Funktionen kann jedes Werkzeug über festgelegte Schnittstellen als PlugIn installiert werden. Somit ist eine nahezu beliebige Erweiterbarkeit der WebSphere Studio Workbench sichergestellt.

Ebenfalls interessant ist, dass die WSW selbst in Java entwickelt wurde. Daraus ergeben sich natürlich einige Vorteile, wie beispielsweise die Unabhängigkeit des Werkzeugs von der Zielplattform. Um die mit der Verwendung von Java-Oberflächen oft einhergehenden Performance-Probleme zu umgehen, hat sich die IBM beim WSW entschlossen, zur Erstellung der grafischen Benutzungsoberfläche eine eigens entwickelte Widget-Klassenbibliothek (Standard Widget Toolkit, SWT) zu verwenden.

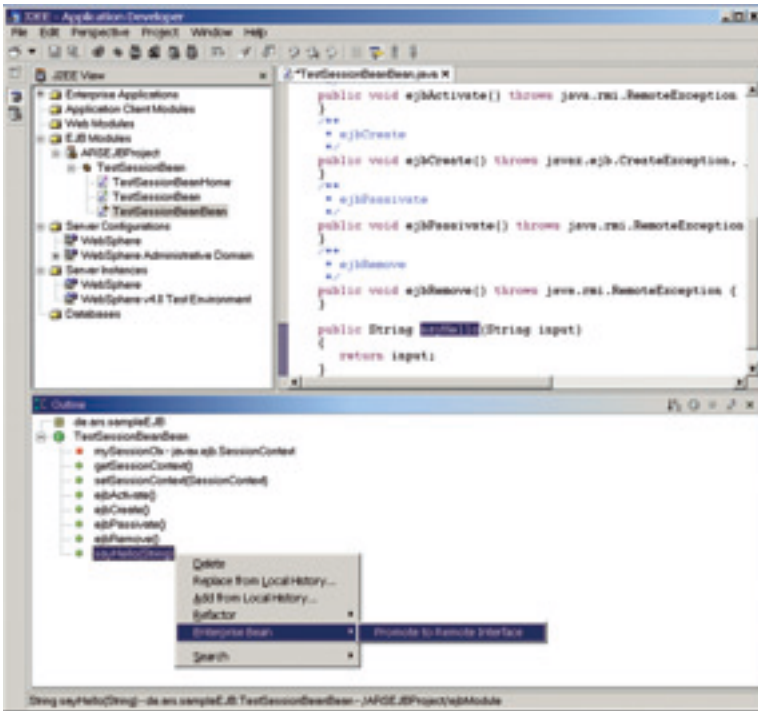


Abb. 4: Übernahme von Methoden in das Remote Interface der EJB

relativ schwierig, da laut Aussage der IBM zunächst kein Ersatz für den zuvor erwähnten Visual Composition Editor (VCE) vorhanden sein wird. Dieser Einschränkung ist sich die IBM natürlich bewusst und hat deshalb in mehreren Gesprächen angedeutet, dass der WSAD in der Version 2 um ein entsprechendes Werkzeug zur Entwicklung von grafischen Benutzungsoberflächen erweitert wird. Gemäß Zeitplan der IBM wird diese neue Version etwa Mitte 2002 verfügbar sein.

WSAD in der Praxis

Im Folgenden betrachten wir anhand der Entwicklung einer beispielhaften Session EJB die neue Entwicklungsumgebung WebSphere Studio Application Developer etwas näher. Beim Starten der IDE wird der langjährige VisualAge Benutzer zunächst wahrscheinlich etwas orientierungslos sein, da die Benutzungsoberfläche völlig neu gestaltet wurde. Die Navigation erfolgt hauptsächlich über die sogenannten Perspectives, die unterschiedliche Sichtenweisen auf die Java- und sonstigen Ressourcen bieten; lax ausgedrückt sind sie nur besonders gestaltete Filter. Nach den ersten Gehversuchen weicht die anfängliche Skepsis allerdings recht schnell der Begeisterung über die neuen Arbeitsabläufe. Außerdem trifft der Benutzer an sehr vielen Stellen auf gewohnte Dialoge und Konzepte. Dies beginnt schon mit der obligatorischen Erstellung eines Projektes, mit welchem sodann alle weiteren Elemente assoziiert werden. Allerdings unterscheidet der WSAD zwischen einer Reihe von unterschiedlichen Projekttypen, wie etwa EJB, J2EE, Java, Web oder Simple Project. Je nach Auswahl werden die möglichen Aktionen des Kontextmenüs entsprechend angepasst. Für die Entwicklung von Enterprise JavaBeans beispielsweise erzeugt man also ein EJB-Projekt. Mögliche Parameter bei der Erstellung sind hierbei natürlich der Name des Projekts und der EAR-Datei, der Speicherort und auch der verwendete Klassenpfad (Class Path). Nach diesem Schritt kann der Wizard zur Erstellung von EJBs gestartet werden, der sich im Aufbau sehr stark an seinem Vorgänger

Das Resultat, ein Werkzeug mit schnellen Oberflächendialogen, bestätigt diese Entscheidung.

Die Idee einer einheitlichen Basis für sämtliche Entwicklungswerkzeuge hat die IBM allerdings noch weiter vorangetrieben, indem sie das gemeinsame Framework als freie Software zur Verfügung stellt. Das zugehörige Open-Source-Projekt trägt den Namen Eclipse. Auf diese Weise soll unter anderem erreicht werden, dass sowohl Partner als auch Fremdhersteller Zusatzwerkzeuge anbieten, welche sich dann optimal in die Produktfamilie integrieren. So wie es aussieht, hat die IBM in diesem Punkt auch ihr Ziel erreicht, da sich zahlreiche Hersteller dieser Idee bereits angenommen haben und entsprechende Zusatzwerkzeuge entwickeln. Eine vollständige und aktuelle Liste der Anbieter finden Sie in den am Ende des Artikels aufgelisteten Quellen. Nach den derzeitigen Informationen werden folgende Hersteller Zusatzwerkzeuge für die WebSphere Studio Workbench anbieten.

- Versata (regelbasierte Anwendungsentwicklung)
- Extricity (Konnektoren)

- Instantiations (zusätzliche entwicklungsunterstützende Werkzeuge)
- Holosofx (Modellierung von Geschäftsprozessen)
- Rational (Requirements Management, Datenmodellierung, Anbindung an Rational ClearCase / ClearQuest, Testwerkzeuge)
- Interwoven (Web Content Management)
- Merant (Anbindung an PVCS)
- Serena (Anbindung an Serena ChangeMan)
- Sitraka (Performance- und Testwerkzeuge)

Auf der WebSphere 2001 Konferenz in Wien wurde von Seiten der IBM sogar angedeutet, in Zukunft auch andere Entwicklungsumgebungen (COBOL, PL/I, MQSeries Integrator) auf die WebSphere Studio Workbench zu standardisieren.

Wie aus der bisherigen Beschreibung des WSAD hervorgeht, ist das Nachfolgeprodukt von VisualAge for Java bisher hauptsächlich auf die Erstellung von serverseitigen Anwendungen zugeschnitten. Die Entwicklung von oberflächenbasierten Anwendungen gestaltet sich hingegen in der Version 1 des WSAD als

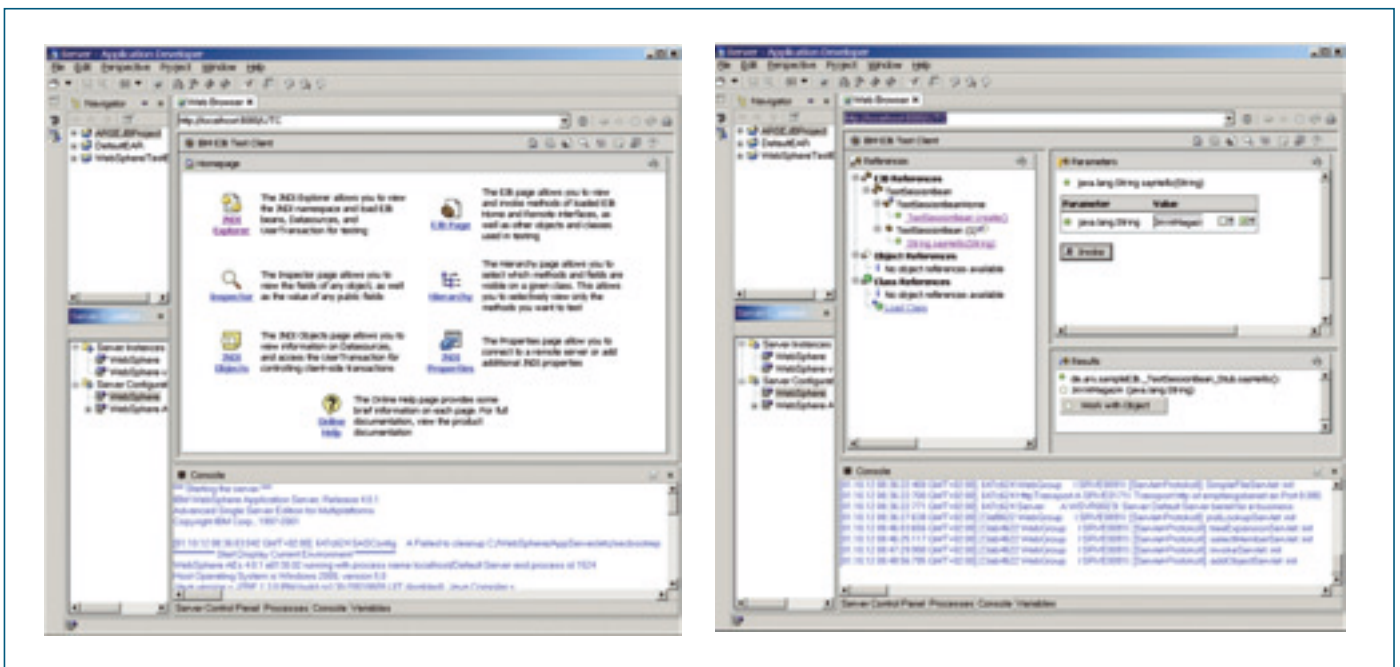


Abb. 5: Neu gestalteter Test-Client für Enterprise JavaBeans

aus VisualAge for Java orientiert (siehe Abb. 3).

Auch im Weiteren ist die Vorgehensweise denkbar vertraut. Dank der Unterstützung durch das Werkzeug kann sich der Programmierer, wie bei VisualAge for Java, voll und ganz auf die Implementierung seiner Business-Logik konzentrieren. Alle gewohnten Hilfsfunktionen wie etwa die automatische Code-Ergänzung stehen im Editor-Fenster zur Verfügung. Neben der kompletten Anzeige der Quelltexte kann auch ein Modus gewählt werden, in dem nur das ausgewählte Element angezeigt wird.

Nach der Übernahme der einzelnen Methoden aus der Bean-Klasse in das Remote Interface (siehe Abb. 4) werden noch die notwendigen Klassen für den Betrieb in der WebSphere Testumgebung (WebSphere Application Server 4.01) generiert („Deploy code and RMIC Stub and Tie code for this bean“). Damit ist die Entwicklungsarbeit abgeschlossen. In einem separat zu erzeugenden Server-Projekt können nun Server-Instanzen und zugeordnete Konfigurationen angelegt werden, wobei zwischen einigen Tomcat Servlet Engines und WebSphere-Ausprägungen gewählt werden kann. Nach dem Start einer EJB in der Testumgebung zeigt

sich ein neu gestalteter Test-Client, in dem die einzelnen Schritte, angefangen bei JNDI-Lookup über die Objekt-Instanziierung bis hin zu entfernten Methodenaufrufen, ausgetestet werden können (siehe Abb. 5)

Fazit

Version 4 wird die letzte Version des erfolgreichen Entwicklungswerkzeuges Visual Age for Java sein. So verwundert es nicht, dass es im Vergleich zur Vorgängerversion nur wenig Neuerungen erfahren hat. Mit dem WebSphere Studio Application Developer stellt die IBM allerdings schon jetzt ein sehr vielversprechendes Nachfolgeprodukt vor, welches neben der erweiterten Funktionalität vor allen Dingen wegen des neuartigen Konzepts für Anwender und Hersteller gleichermaßen interessant ist. Man darf also gespannt sein, ob sich die Idee eines gemeinsamen Frameworks für sämtliche Java-Entwicklungswerkzeuge in Zukunft durchsetzen und den Markt für Entwicklungsumgebungen revolutionieren wird.

Stefan Schäffer und Walter Schilder sind Leiter des Unternehmensbereiches Application Development der ARS Computer und Consulting GmbH in München (www.ars.de).

Links & Literatur

- Informationen zu VisualAge for Java
- www-4.ibm.com/software/ad/vajava/
- www7.software.ibm.com/vad.nsf/Data/Document2020
- www-4.ibm.com/software/ad/vajava/about/v40/vaj40faq.html#15

Informationen zu den auf der WebSphere Studio Workbench basierenden Produkten

- www-4.ibm.com/software/ad/workbench/about/
- www-4.ibm.com/software/ad/workbench/about/

Informationen zum Open-Source-Projekt Eclipse

- www.eclipse.org

ARS

ARS
Computer und Consulting GmbH
Ridlerstraße 55
80339 München
Telefon 089/32468-00
Telefax 089/32468-288
info@ars.de