KAP DION

.NET Dlls in Java-Webapplikationen

Integration von Dotnet Dlls in Java-Webapplikationen mittels JNBridge

Autoren:

Harald Negrin	IT-Leiter BAWAG-Versicherung
Jens Bühring	Geschäftsführer Kap Dion – Gesellschaft für Bankensoftware, Wien

erschienen in Java-Magazin, Frankfurt am Main, 02 2008, Seite 104-106

Integration von Dotnet Dlls in Java-Webapplikationen mittels JNBridge

In der Versicherungs-IT ist es Stand der Technik, so genannte Produktrechenkerne, welche die Logik eines Versicherungsproduktes abbilden, in verschiedenen Softwareumgebungen zu nutzen. Die Rechenkerne werden beispielsweise in die Angebotssoftware auf den Laptops der Vertriebsmitarbeiter eingebunden und gleichzeitig auf Servern für die laufende Verwaltung der bestehenden Versicherungsverträge verwendet. Diese Mehrfachnutzung senkt die Entwicklungskosten und garantiert, dass die Berechnung in verschiedenen Systemen immer exakt gleich abläuft.

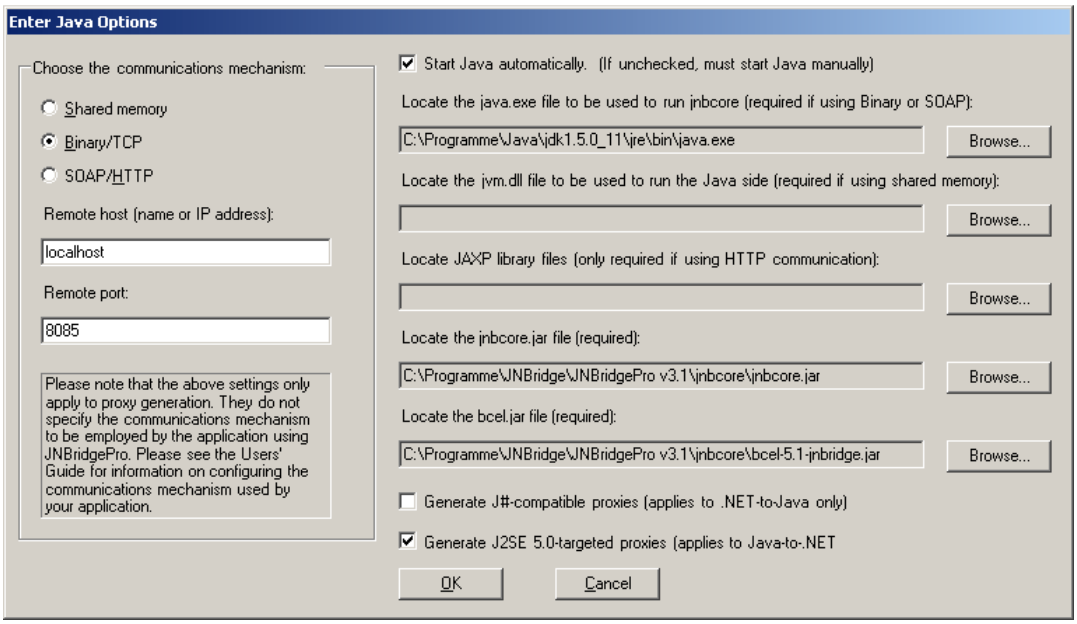
Im konkreten Fall liegt der Rechenkern in Visual-Basic vor und hat sich schon in einer Arbeitsplatz-Angebotssoftware bewährt. Nun soll er mit Hilfe der Lösung JNBridge in eine Java-Server-Faces-Webapplikation integriert werden.

Schnelle Verbindung zwischen Java und Dotnet

Der Visual-Basic Rechenkern liegt für das Projekt als Dotnet 1.1 Dll vor. Für die Einbindung wird die kommerzielle Lösung JNBridge verwendet. Mit dieser Lösung ist es einfach, den Rechenkern in das Java-Projekt einzugliedern, ohne sich näher mit der Dotnet-Seite zu beschäftigen. JNBridge stellt 3 Kommunikationswege zwischen den so genannten Java-Proxy-Klassen und den Dotnet-Klassen zur Verfügung: über TCP Protokoll, über JNI oder über Webservices.

© Kap Dion 15. November 2009

Festlegung der Kommunikation zwischen Java und Dotnet als TCP (Abb. 1)

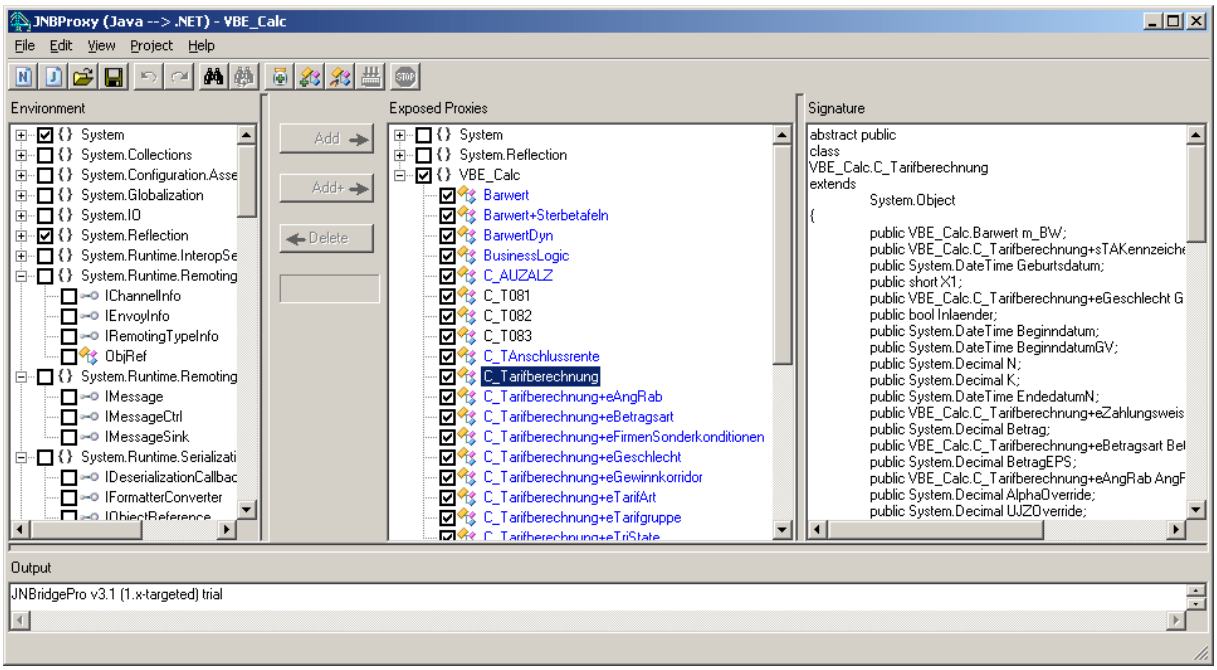


Für die Verknüpfung der beiden Teile der Applikation wählen wir „Binary/TCP“, da die Applikation verteilt auf einem Windows-Server und einem Solaris-Server laufen soll. Da kein öffentlicher Webservice zu erstellen ist, findet anstelle von „SOAP/HTTP“ das laut Hersteller schnellere „Binary/TCP“ Protokoll Verwendung.

Von dem Rechenkern ist nur die Programmierschnittstelle bekannt. Diese Schnittstelle nutzt Enumerations sowie komplexere eigene Datentypen und besitzt ca. 30 Eingabevariablen sowie 20 Ergebnisvariablen, darunter einige Arrays. Ein schönes Feature von JNBridge ist, dass die Klassen der Dll aufgelistet werden und die für die Schnittstelle benötigten Klassen mit einem Klick ausgewählt werden können.

© Kap Dion 15. November 2009

Auswahl der Klassen für die Schnittstelle (Abb. 2)



JNBridge generiert zu den Dotnet-Klassen so genannte Proxy-Klassen in Java. Diese liegen als Java-Library vor, welche in den Klassenpfad aufgenommen wird. Die Proxy-Klassen können im Java-Code wie jede andere Klasse eingebunden werden. Die Java-Proxy-Klassen rufen über TCP die eigentlichen Klassen der Dll auf.
Obwohl die Schnittstelle des Rechenkerns einige Komplexität mit sich bringt, lässt er sich mit Hilfe der Code-Generierung innerhalb kurzer Zeit in das Projekt einbinden.

Probleme mit der Datenübertragung

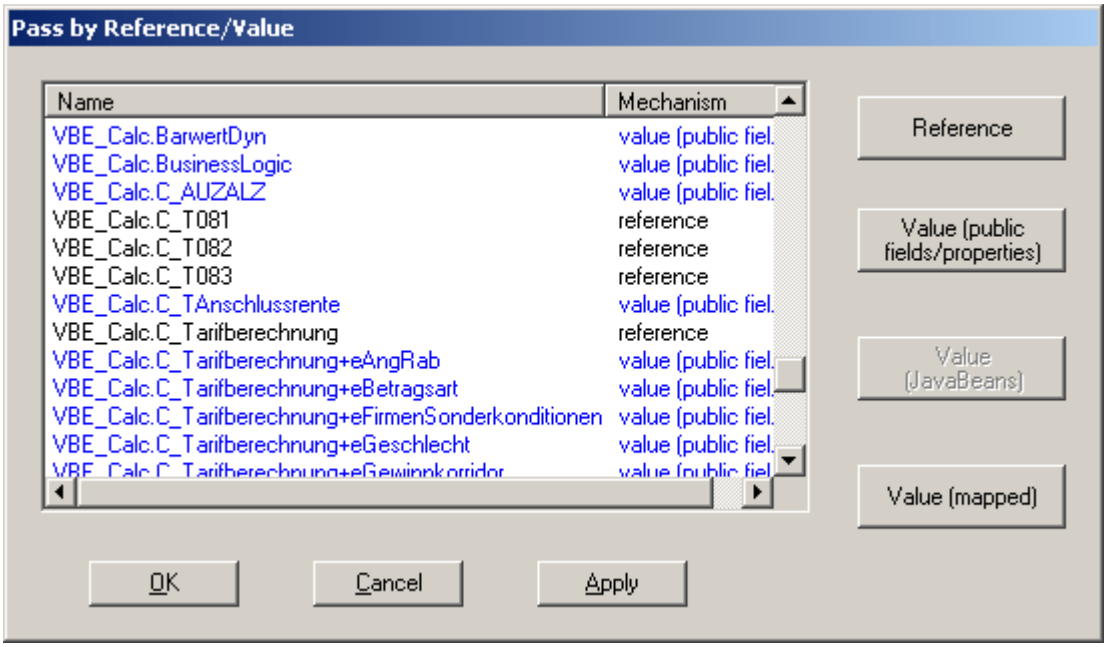
Ein verteilter Testaufbau, bei dem Dotnet-Seite und Java-Webapplikation auf zwei verschiedenen Servern installiert sind, offenbart gravierende Performance-Probleme: Anstatt die Ergebnisseite der Webapplikation anzuzeigen, meldet der Browser einen Time-out. Nach Einbau entsprechender Log-Meldungen rund um die Berechnung zeigt sich, dass die Berechnung inakzeptable 3 Minuten dauert.

Ursache hierfür ist, dass das Thema Datenübertragung bisher noch nicht explizit behandelt wurde. Bei der ersten Generierung des Proxys waren, ohne dass dieser Punkt besondere Berücksichtigung fand, alle Objekte als „remote“ festgelegt worden, d.h. die Objekte wurden auf dem Windows-Server an die Session gebunden und dort geführt. Jedes Einzelergebnis muss in diesem Fall für die Darstellung am Windows-Server abgeholt werden. Da es wegen

der Arrays relativ viele Werte sind, führt dies pro Berechnung zu ca. 450 Aufrufen über das Netzwerk. Diese Aufrufe übermitteln oft nur den Wert einer einzelnen Variable, aber ihre große Zahl und die Herstellung der zahlreichen TCP-Verbindungen führt zu extrem langen Aufrufzeiten der Dll.

Nun bietet JNBridge die Möglichkeit für jede Klasse festzulegen, auf welcher Seite die Werte gespeichert werden sollen. Gleichzeitig lässt sich damit spezifizieren, wie die Datenübertragung abläuft.

Beispiel, wie das Handling der Klassen in unterschiedlicher Weise festgelegt wird (Abb. 3)



Eine verbesserte Lösung

Gewünscht ist, dass nach der Berechnung das gesamte Ergebnis in einem Schritt zur Webapplikation übertragen wird. Eine Umkonfiguration der Schnittstelle auf lokale Objekte erweist sich im konkreten Fall aber als unmöglich.
Da der Rechenkern ursprünglich für eine lokale Arbeitsplatzapplikation und nicht für einen Remote-Aufruf konzipiert wurde, gibt es in der Dll keine Objekte, die für eine gemeinsame Datenübertragung aller Eingaben bzw. aller Ergebnisse geeignet sind. Es ist daher unumgänglich, auf der Dotnet-Seite einzugreifen und zunächst geeignete Objekte für eine Datenübertragung zu schaffen. Da die Rechenkern-Dll in der Verantwortung der Fachabteilung liegt und von dem Projekt unberührt bleiben soll, muss eine neue Wrapper-Dll

erstellt werden. Diese enthält spezielle Objekte für die Datenübertragung und greift für die Berechnung auf den eigentlichen Rechenkern zurück.
Zu dieser neuen Wrapper-Dll wird wiederum mit Hilfe von JNBridge ein zugehöriger Java-Code generiert. Jetzt können die neuen Datenobjekte für Input und Output der Berechnung so konfiguriert werden, dass sie als ganzes Objekt übertragen werden. Die Zahl der Aufrufe sinkt damit auf zwei: einen Aufruf zum Übermitteln des Input-Objekts und zum Starten der Berechnung und einen weiteren zum Abholen des befüllten Ergebnisobjekts. Die Berechnungszeit beträgt dann nur noch eine Sekunde.

Deployment und Betrieb

Da Java- und Dotnet-Seite mittels TCP-Protokoll verbunden sind, besteht die Möglichkeit, die Webapplikation in zwei Teilen zu hosten: Die Java-Webapplikation kann auf einer Sun-Solaris untergebracht werden, während der Dotnet-Rechenkern und die JNBridge-Software auf einem Windows-Server installiert werden. Dies hat den Vorteil, dass die gewohnte Infrastruktur für die Webapplikation genutzt werden kann. Die beiden Server stehen in verschiedenen Sicherheitszonen, so dass zusätzlich eine Firewall freizuschalten ist. Wäre JNI eingesetzt worden, müssten beide Teile der Applikation auf einer Windows-Maschine betrieben werden. Im konkreten Fall hätte die für Windows-Systeme zuständige Abteilung Verantwortung für einen Java-Servlet-Container neu übernehmen müssen.

© Kap Dion 15. November 2009

Fazit

Die Einbindung von Dotnet Dlls ist mit Hilfe von JNBridge kein schwieriges Unterfangen. Gerade wenn die Dll eine umfangreiche API bereitstellt, ist die kommerzielle Lösung JNBridge hilfreich, da über Codegenerierung schnell die entsprechende Java-API zur Verfügung steht.
Die Dokumentation ist allerdings nur auf Englisch verfügbar und aufgrund ihres Umfangs nicht immer ganz übersichtlich. Kompensiert wird dies durch den prompten und engagierten Support des Herstellers.
Die Möglichkeit Java und Dotnet über TCP zu verbinden, ist unter dem Gesichtspunkt des Betriebs attraktiv: Der Java-Teil der Applikation kann dann (weiter) auf einem UNIX-System betrieben werden.
Bei einer solchen Remote-Verbindung erspart JNBridge nicht die explizite Auseinandersetzung mit dem Thema der Datenübertragung. Sind wie in unserem Beispiel auch auf der Dotnet-Seite spezielle Containerobjekte für eine effiziente Datenübertragung anzulegen, reduziert sich der Vorteil von JNBridge. In einem solchen Fall kann auf der Dotnet-Seite auch gleich ein Webservice erstellt werden.

JNBridge	www.jnbridge.com
Java Magazin	www.javamagazin.de
BAWAG-P.S.K. Versicherung AG	www.bawagpskvers.at
Kap Dion	www.kapdion.com

© Kap Dion 15. November 2009