Management von J2EE-Applikationen mit JMX

Conference Proceeding Paper

Dr. Gregor Karl Frey, SAP

Gregor Frey studierte Philosophie an der Universität Heidelberg. 1996 kam er zu SAP. Dort war er zunächst im Bereich SoftwareLogistik tätig. Als Development Architect ist er momentan für Monitoring und Management von Java Applikationen verantwortlich und beschäftigt sich unter anderem mit der Integration von JMX in den SAP WebApplicationServer.

Thema

„JMX" steht für „Java Management Extensions“ und soll dem Management von Java Applikationen eine einheitliche Infrastruktur bescheren. Mit der Version 1.4 wird JMX offizieller Bestandteil von J2EE. Prinzipiell sollte man erwarten, daß Applikationen, Server und Management- Tools von verschiedenen Herstellern beliebig kombiniert werden können. Die Realität sieht allerdings anders aus. Der Umfang und die Integration von JMX kann je nach J2EE-Server ganz unterschiedlich ausfallen, so daß der Portierungsaufwand für eine einmal entwickelte Lösung beachtlich nach oben geschraubt wird.

Zunächst gebe ich eine kurze Einführung in JMX und in das Management eines J2EE-Servers, soweit es durch Spezifikatione abgedeckt ist. Die Java Management Extensions (JSR 3) definieren die Architektur, die Design Patterns, die APIs, und einige Services für das Applikations- und Netzwerk Management in Java. Die JMX Spezifikation gibt Java-Entwicklern aus allen Industriezweigen die Mittel um Java Code zu instrumentieren, Java Agenten zu erzeugen, verteilte Management -Middleware und -Konsolen zu implementieren, und diese Lösungen nahtlos in bestehende Management Systeme zu integrieren.

Orthogonal zu JMX definiert die Spezifikation zum J2EE Management (JSR 77) ein abstraktes Informationsmodell, das leicht an jedes Management Protokoll angepaßt werden kann. Im JSR 77 wird festgelegt, welche gemanagte Objekte in einem J2EE Server überhaupt existieren, wie sie miteinander verbunden sind und welche Eigenschaften sie haben. Die API für den Zugriff auf dieses Informationsmodell wird als ein remote Interface einer Enterprise Java Bean spezifiziert. Diese EJB heißt „J2EE Management EJB Component“ oder auch kurz MEJB.

In gewisser Weise steht die MEJB, anders als das abstrakte Datenmodell des JSR 77, in direkter „Konkurrenz“ zu den von JMX definierten APIs. In beiden Fällen geht es um die Schnittstellen für den Zugriff auf Monitoring- und Management-Funktionen. Es handelt sich aber nicht wirklich um eine Konkurrenz. Vielmehr setzt die MEJB auf die JMX-APIs auf und liefert eine remote Schnittstelle, die im ursprünglichen JSR 3 noch ausgeklammert wurde und erst mit dem JSR 160 festgelegt werden wird.

In der Version 1.4 der J2EE-Platform (JSR 151) ist die Spezifikation des J2EE Managements gemäß JSR 77 enthalten. Auch JMX gehört zu den API-Anforderungen von J2EE 1.4, aber mit Einschränkungen:

„The only JMX support required is specified in the J2EE Management specification.“

Die J2EE Management Spezifikation ihrerseits verlangt:

„The MEJB component exposes the manageable resources on a J2EE platform as JMX Managed Beans (MBeans) and requires an implementation of the JMX public APIs specified by the Java Management Extensions Instrumentation and Agent Specification, v1.1.“

Was aber diese „public APIs“ sind, wird nicht ausgeführt und war auch auf Nachfragen nicht in Erfahrung zu bringen. Für den Umfang und die Integration von JMX bleibt damit als Fazit, daß man beim Grad der JMX-Unterstützung bei J2EE 1.4 konformen Applikationsservern mit einer ziemlichen Bandbreite rechnen muß.

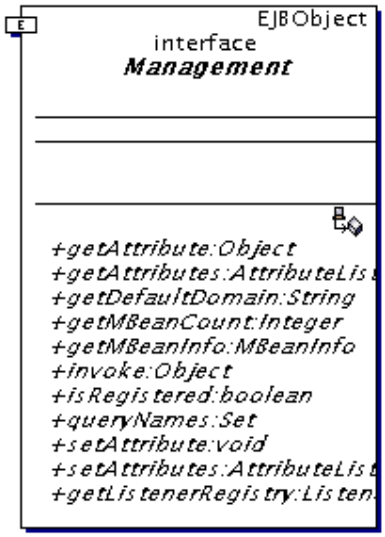
JMX im SAP Web Application Server

Im folgenden werde ich nun darstellen, in welcher Weise JMX in SAPs J2EE-Server, dem SAP Web Application Server, integriert wurde. Meine Ausführungen beziehen sich dabei auf die Version 6.30, die vorraussichtlich im dritten Quartal dieses Jahres auf den Markt kommt.

Der SAP WebAS enthält ab dem Release 6.30 eine vollständige JMX Implementierung in der Version 1.2. Vorraussichtlich wird diese JMX Implementierung bis zum Erscheinungstermin des WebAS offiziell zertifiziert worden sein.

Allein, die standard-konforme Implementierung von JMX läßt noch völlig offen, in welcher Art JMX im Server selbst benutzt wird und welchen Zugriff der Server Applikationen auf die JMX Imfrastruktur gibt.

Betrachten wir zur Verdeutlichung die Frage, auf welche Weise die MEJB des JSR 77 realisiert werden kann. Dazu seien hier die Methoden der MEJB und des in JMX definierten MBeanServers gegenübergestellt.



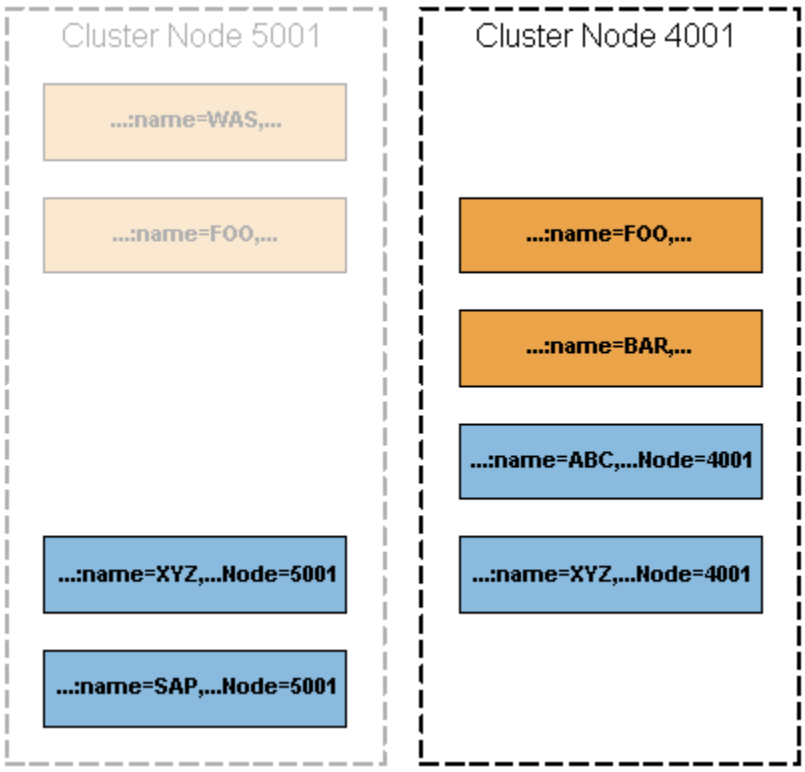
Man sieht, daß das Management Interface im wesentlichen eine Teilmenge des JMX MBeanServers definiert. Es ist deshalb naheliegend die MEJB als Fasade eines MBeanServers zu implementieren, bei dem alle gemanagten Resourcen des J2EE-Servers als MBeans registriert werden. In diesem Szenario würde JMX als durchgehende Infrastruktur zum Management des Servers verwendet.

Die JSR 77 Spezifikation läßt aber zu, daß die MEJB auch ganz ohne MBeanServer implementiert werden kann. Nach außen macht dies keinen Unterschied. Intern würde das aber bedeuten, daß der J2EE-Server noch nicht einmal einen MBeanServer instantiieren muß, geschweige denn, daß er eine durchgehende Instrumentierung gemäß JMX besitzt.
Fazit: Die Existenz eines MBeanServers in der VM des J2EE-Servers kann nicht vorausgesetzt werden.
Die J2EE Spezifikation läßt darüber hinaus zu, daß selbst für den Fall, daß im J2EE Server ein MBeanServer instantiiert wird, Applikationen keine Berechtigung zum Zugriff auf den internen MBeanServer haben. Für Applikationen ist es aber keine gute Lösung, wenn sie jeweils eigene MBeanServer starten, weil für Monitoring und Management die Verwendung eines gemeinsamen MBeanServers sinnvoller ist und das Starten eines MBeanServers aus einem Servlet oder einer EJB wegen deren spezifischen Lifecycles problematisch ist.
Im SAP Web AS 6.30 ist die MEJB auf Basis eines MBeanServers implementiert. Dieser interne MBeanServer des SAP WebAS steht Services und Applikationen zur Verwendung offen, vorausgesetzt, daß die Applikation über die entsprechenden Berechtigungen verfügt.
Mit der Verwendung des internen MBeanServers haben alle Appliaktionen und Services Zugang zur kompletten JMX-Infrastruktur im WebAS. Auf die folgenden Merkmale der SAP JMX-Infrastruktur werde ich nun im einzelnen eingehen:

- Clustering
- Connectoren
- ClientAPI
- Anzeige

Clustering

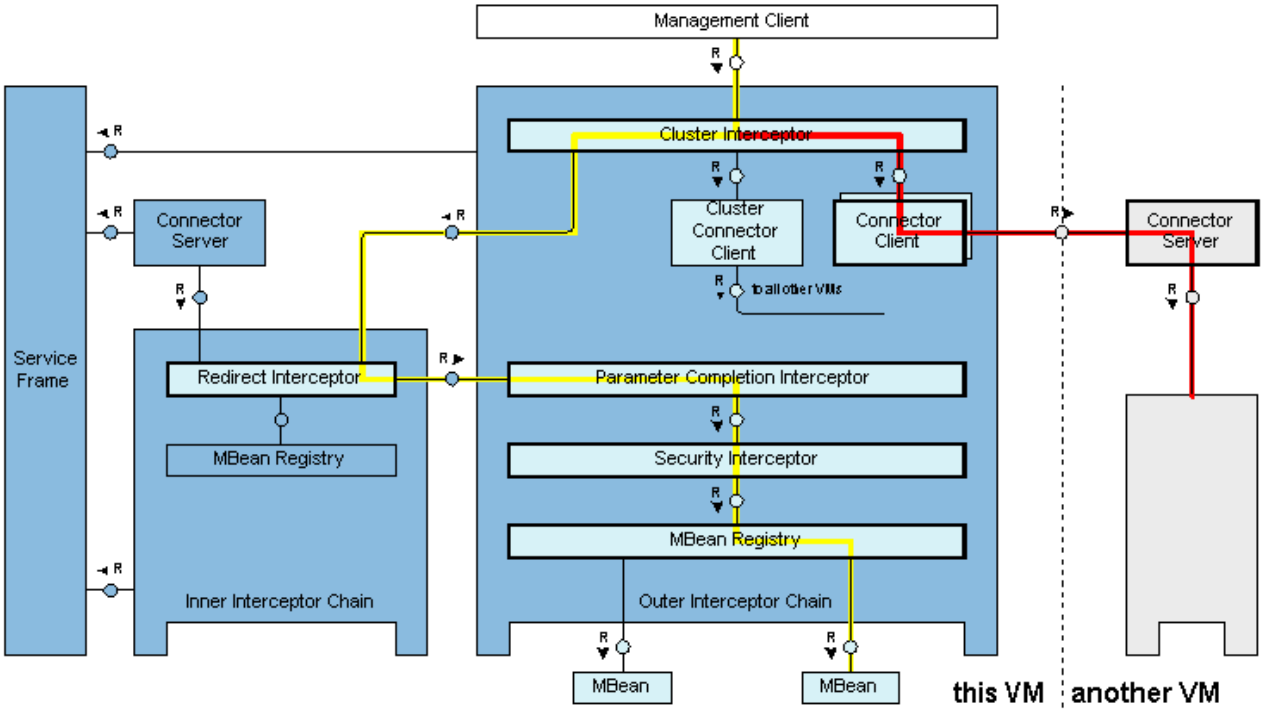
Um eine bessere Skalierung zu erreichen besteht der SAP WebAS aus einem Cluster von J2EE-Server Prozessen. In jedem J2EE-Server Prozess gibt es genau ein MBeanServer, der für die MBeans der jeweiligen VM zuständig ist. Prinzipielle ist die Registrierung einer MBean bei einem MBeanServer immer lokal, d.h. MBeanServer und registrierte MBeans leben immer in der selben Virtual Machine.
Auf der anderen Seite muß es aber auch möglich sein, MBeans cluster-weit sichtbar zu machen. Denn da von vorneherein gar nicht feststeht, auf welchem Cluster-Knoten eine Applikation tatsächlich instantiiert wird, fehlt einem externen Client die Möglichkeit, sich gezielt an den MBeanServer des entsprechenden Knotens zu konnektieren. Normalerweise landet ein externer Cleint über einen Connector in irgendeinem Cluster Prozess. Wie erreicht der Client von hier aus die MBeans der anderen Prozesse?
Im SAP WebAS werden drei Sorten von MBeans unterschieden:
Local MBeans existieren jeweils nur in einem einzigen Server Prozess, sind aber von allen anderen Prozessen aus sichtbar. Um sie zu unterscheiden enthalten sie in ihrem Namen die Knoten ID.
Clustered MBeans enthalten dagegen keine Knoten ID in ihrem Namen. Sie existieren synchronisiert in allen Prozessen.
Internal MBeans existieren in einem einzigen Prozess und sind auch nur lokal sichtbar. Auch sie enthalten keine KnotenID im Namen.



Im dargestellten Beispiel sind für einen externen Client, der sich an den Knoten 4001 konnektiert hat, insgesamt 6 MBeans sichtbar: ABC, XYZ (auf Knoten 4001 und auf Knoten 5001) und SAP sind local MBeans. FOO ist eine clustered MBean und BAR ist eine internal MBean. Die ersten fünf der genannten MBeans wären auch vom Knoten 5001 aus sichtbar. Die interne MBean BAR dagegen nicht.

Connectoren

Neben der im JSR 77 spezifizierten MEJB enthält der SAP WebAS eine JSR 160 konformen MBeanServerConnection. Dieser Connector gibt die komplette MBeanServer Schnittstelle über RMI an externe Clients weiter. Er trägt Sorge dafür, daß die Methodenaufrufe immer an die „richtigen“ Cluster-Knoten weitergegeben werden.



Der SAP WebAS benutzt eine InterceptorChain, um die an den MBeanServer gerichteten Methoden Aufrufe abzufangen und an die „richtigen“ Server im Cluster weiterzuleiten
Beispiel 1:

- Es soll ein Attribut einer MBean gesetzt werden, die in einem anderen Prozess zu Hause ist (local MBean).
- Der ClusterInterceptor erkennt anhand der NodeID, welches der zuständige Server ist.
- Der ConnectorClient leitet den Call an den ermittelten Server weiter.

Beispiel 2:
Über eine Query sollen in allen Prozesse die MBeans eines bestimmten Typs gefunden werden

- Der ClusterInterceptor erkennt an der fehlenden NodeID, daß es sich um eine Query an alle Prozesse handelt.
- Der ClusterConnectorClient leitet den Call an alle Server weiter.
- Und faßt die Ergebnismengen zusammen.

Client API

Die Client API der JMX-Infrastruktur im SAP WebAS gibt zwei Möglichkeiten zur Durchführung eines Lookups eines Connectors: Einen JSR 160 konformen Lookup und einen direkten Lookup über JNDI.
Hier ein Beispiel für einen JSR 160 konformen Lookup:

```
JmxClient clientAPI = JmxClientFactory.getJmxClient();
JmxConnector connector =
clientAPI.getConnector("jmx:sapj2ee:4001");
MBeanServerConnection server =
connector.getMBeanServerConnection();
```

Und hier ein Beispiel für einen direkten JNDI Lookup:

```
InitialContext ctx = new InitialContext(props);
obj = ctx.lookup("J2EE_Management/ExtendedMEJB");
ExtendedManagementHome home = (ExtendedManagementHome)
PortableRemoteObject.narrow(obj,
ExtendedManagementHome.class);
MBeanServerConnection server = home.create();
```

Im ersten Fall wird eine Client API benutzt, die per RMI remote eine Verbindung zum Server herstellt. Im zweiten Fall wird vorausgesetzt, daß der Naming Service des WebAS zur Verfügung steht. Die MBeanServerConnection kann in diesem Fall über die für EJBs übliche Verfahren erzeugt werden.
In beiden Fällen ist das Ergebnis das gleiche: Der Client erhält eine Connection, über die er auf den (die) MBeanServer des WebAS zugreifen kann.

Zusammenfassung

Mit der J2EE 1.4 Spezifikation wird JMX als eine Anforderung für konforme J2EE-Server genannt. Die Spezifikation läßt jedoch offen, welche Teile von JMX tatsächlich enthalten sein müssen, und wie es in den Server integriert wird.
Im SAP WebAS wird JMX 1.2 vollständig in den Server integriert.
Zusätzlich enthält der WebAS viele JMX-Features, die durch die Spezifikation nicht erfordert werden: Clustering, Connectoren, ClientAPI und viele weitere Features auf die hier nicht mehr eingegangen werden kann.