

# ERFOLGREICHERE JAVA-ANWENDUNGEN –alles, was Sie wissen müssen

JAVA-ANWENDUNGSMANAGEMENT UND J2EE

# Inhalt

Einleitung .....	2
Browser .....	2
Webserver .....	3
JVM .....	3
Servlets .....	4
Enterprise Java Beans .....	4
Java Message Service .....	5
J2EE-Managementschnittstelle .....	5
Zusammenfassung und Empfehlungen .....	6
Wir helfen Ihnen, Ihren Wettbewerbsvorteil zu wahren .....	6

## EINLEITUNG

Für Unternehmen, die Anwendungen eines übernommenen Unternehmens integrieren, eine Schnittstelle zwischen Anwendungen und Internet einrichten oder auf eine Legacy-Datenbank zugreifen müssen, ist J2EE eine leistungsstarke Lösung, die die traditionellen Standards der Anwendungsprogrammierung nutzt. Wie aber finden Sie unter all den Standards diejenigen, die Sie wirklich benötigen? Stellen Sie sich J2EE einfach als unternehmensweites Java vor, wobei das Unternehmen eine gemischte Plattform aus allerlei Technologie ist, die die Geschäftsprozesse des Unternehmens stützt. Java kann in dieser Umgebung der gemeinsame Nenner sein. Es ist die Programmiersprache, mit der Sie einzelne Bausteine erstellen, die dann zu einer Anwendung zusammengesetzt werden können. Java hat einen ganz wesentlichen Vorteil: Wenn alle Plattformen dieselbe Sprache sprechen, bleiben dem Unternehmen die Kosten für die Verbindung einzelner Anwendungen auf verschiedenen Plattformen mit Hilfe proprietärer Mittel erspart.

Dieser Standard ermöglicht die Erstellung von Software-Bausteinen (Objekten), die eigenständig, funktionell und auf all den verschiedenartigen Plattformen des Unternehmens wiederverwendbar sind. Konkret bedeutet dies, dass ein Unternehmen schneller Anwendungen entwickeln kann, die sowohl die Webseiten, die der Anwender im Browser sieht, umfassen als auch eine ausgefeilte Anwendungsebene oder ein Backend. Die J2EE-Anwendungsebene kann problemlos auf Daten in einer beliebigen Datenbank zugreifen und mit vielen verschiedenen Arten von Middleware-Nachrichtendiensten interagieren. Welche Serverhardware-Plattformen dafür eingesetzt werden, ist für die Anwendung weitgehend irrelevant. Daher ist J2EE bestens geeignet, wenn Sie verschiedenartige Anwendungen miteinander verbinden und ineinander integrieren möchten.

Damit Sie J2EE erfolgreich einsetzen können, müssen Sie die verschiedenen Phasen und Komponenten einer typischen Webanwendung verstehen. Einige Branchenanalysten haben die Evolution der Computeranwendungen mit der des Autos verglichen. Heute sind Autos eine hochkomplexe Sammlung genauestens aufeinander abgestimmter Komponenten, die alle mit dem Ziel zusammenarbeiten, die Räder zu bewegen. Mit der erfolgreichen J2EE-Anwendungsentwicklung verhält es sich ähnlich. Oberflächlich betrachtet, unterstützt Java die Integration und ermöglicht die schnelle Bereitstellung des gewünschten Business-Service. J2EE erleichtert diese Arbeit zwar, doch unter der Oberfläche stützt es sich

auf zahlreiche Bausteine, die koordiniert werden müssen und Spitzenleistung erfordern. Wenn Administratoren und Betreiber beobachten, wie gut oder schlecht diese Bausteine funktionieren, können sie eine vorhersagbare und zuverlässige Produktionsumgebung schaffen.

## BROWSER

Die meisten Menschen greifen heute über einen Browser in einem Netzwerk auf das Internet zu. Das Problem dabei ist, dass die meisten Java-Programmierer voraussetzen, dass den von ihren Programmen erzeugten Browserseiten unbegrenzte Netzwerkbandbreite zur Verfügung steht. Doch nicht alle Netzwerke verfügen über eine unbegrenzte Bandbreite (fragen Sie nur jemanden, der am Flughafen über einen PDA auf das Internet zugreifen möchte). Die Folge sind lange Reaktionszeiten des Browsers. Wer definiert aber, was eine „lange“ Reaktionszeit ist? Gilt die Definition des Browser-Nutzers oder die des Java-Programmierers? Die Anwendungs- und Betriebsteams müssen definieren, was eine lange oder eine kurze Reaktionszeit ist. Wie kann dann aber die Reaktionszeit des Browsers tatsächlich erfasst werden?

## WEBSERVER

Sobald die Reaktionszeit des Browsers quantifiziert, erfasst und gemessen wurde, müssen Sie Ihre HTTP-Server (auch als Webserver bezeichnet) genauer betrachten. Diese Server entscheiden, wie Ihre Transaktion abgewickelt wird. Wird nur eine einfache Datei angefragt (zum Beispiel eine statische HTML-Seite), so ruft der Webserver die Seite wahrscheinlich selbst ab und gibt sie an den Browser zurück. Wenn die Transaktion dynamisch erstellten Inhalt (zum Beispiel den aktuellen Kontostand) umfasst, gibt der Webserver die Anfrage eventuell an den Webanwendungsserver weiter. Dort befinden sich die J2EE-Komponenten – Servlets, Java Server Pages (JSP) und Enterprise Java Beans (EJB). Bevor die Transaktion jedoch an J2EE übergeben wird, sehen wir uns etwas genauer an, was der Webserver macht.

Die an einen Browser geschickten Webserver-Seiten sind aus vielen verschiedenen Teilen zusammengesetzt: Textseiten, Bilder und Client-Anwendungen (etwa Java-Applets). Wenn die Reaktionszeit auf Anwenderseite lang ist, wie können Sie dann erkennen, welche Teile der Seiten dafür verantwortlich sind? Wenn Sie wissen, welche Bestandteile der Browserseiten Probleme bereiten, können Sie die Problembeseitigung beschleunigen. Wir gehen nun davon aus, dass auf dem Webserver alles bestens läuft und die J2EE-Transaktion an ein Servlet auf der JVM übergeben wird.

## JVM

Servlets koordinieren die Gesamtlogik des Programms in einer J2EE Java Virtual Machine (JVM). Bevor wir uns ausführlicher mit Servlets befassen, sollten wir uns die JVM selbst ansehen. Wenn die JVM schlecht läuft oder nicht genau koordiniert ist, fällt natürlich auch die Leistung der Servlets, die auf der JVM ausgeführt werden, schlecht aus.

In Java wird anders programmiert als in vielen anderen Programmiersprachen. In anderen Sprachen wird der Code in Maschinensprache kompiliert, in Java hingegen nicht. Java-Code wird in eine Sprache kompiliert, die dann von der JVM ausgeführt wird. Mit Java kompilierter Code ist nicht nativer Code des Betriebssystems, sondern der JVM. Daher muss die JVM aus zwei Gründen Betriebssystem-Speicher (den so genannten „Heap“) zuweisen:

- > Um die JVM selbst auszuführen
- > Um in der JVM residente Java-Programme auszuführen

Java unterscheidet sich noch in einem weiteren Punkt von anderen Programmiersprachen: Dank der J2EE-Spezifikation braucht sich der Programmierer nicht um die Speicherverwaltung zu kümmern. In traditionellen Programmiersprachen ist die Speicherverwaltung (die richtige Zuweisung und Rücknahme von Speicher) Aufgabe des Anwendungsprogrammierers. In Java ist sie Aufgabe der JVM.

Die JVM startet mit einem anfänglichen Speichervolumen, das vom Betriebssystem des Hosts zugewiesen wird. Dieses Speichervolumen wird gelegentlich als Heap-Größe bezeichnet. Der JVM steht jedoch nicht nur die anfängliche Heap-Größe zur Verfügung; diese ist einfach das Speichervolumen, mit dem die JVM ihre Arbeit beginnt. Im Laufe der Zeit benötigen Java-Anwendungen und die JVM selbst mehr Speicher (mehr als die anfängliche Heap-Größe). Die JVM

weist (in vordefinierten Mengen) mehr Speicher des Host-Betriebssystems zu. Es muss unbedingt überwacht werden, wie schnell und wie viel Speicher des Host-Betriebssystems die JVM zuweist. Wenn die Speicherzuweisung nicht sorgfältig verwaltet wird, steht dem Host-Betriebssystem eventuell nicht mehr ausreichend Speicher zur Verfügung.

Ein weiterer wichtiger Aspekt der Speicher-Heap-Verwaltung ist der freie Speicherplatz, über den die JVM verfügen kann. Dies ist das Speichervolumen (in der Regel in MB angegeben), das die JVM innerhalb des derzeit zugewiesenen Heaps noch verwenden kann. Diese Zahl ist wichtig, da die JVM möglicherweise aussetzt, wenn ihr kein Speicher mehr zur Verfügung steht. Wenn dieser Wert null ist, ist bei der Garbage Collection (mehr dazu weiter unten) eventuell ein Problem aufgetreten. Möglich ist auch, dass eine Java-Anwendung keinen Speicher bzw. keine Objekte freigibt (dies wird auch als Speicherleck bezeichnet).

Wie erkennen Sie den derzeit in der JVM genutzten Prozentsatz der maximal möglichen Heap-Größe? Die JVM weist zunehmend mehr Speicher zu, allerdings nur bis zur maximalen Heap-Größe. Die maximale Heap-Größe wird beim Starten der JVM festgelegt und kann nicht geändert werden, solange die JVM läuft. Es ist wichtig zu wissen, welcher Prozentsatz der maximal möglichen Heap-Größe derzeit verwendet wird.

Manchmal nimmt die Heap-Größe der JVM zu, wenn die Garbage Collection nicht einwandfrei funktioniert. Als Garbage Collection wird der Prozess bezeichnet, mit dem die JVM ungenutzte Speicherbereiche freigibt und der verfügbaren Heap-Größe hinzufügt. Die JVM ist dafür zuständig, freien Speicherplatz zu sammeln und wieder zu verwenden, da die J2EE-Spezifikation den

Java-Programmierer von dieser Aufgabe befreit. Die Garbage Collection hat eine positive und eine negative Seite. Ihr Vorteil besteht darin, dass sie der JVM mehr Speicher für den JVM-Heap zur Verfügung stellt. Sie hat jedoch den Nachteil, dass die JVM während der Garbage Collection J2EE-Transaktionen langsamer abwickelt. Daher ist die Garbage Collection ein Drahtseilakt: Wenn sie nicht stattfindet, geht der JVM irgendwann der Speicher aus. Wenn sie zu oft stattfindet oder zu lange dauert, laufen Java-Anwendungen langsamer. Die Garbage Collection sollte anfangs schnell gehen und dann zunehmend länger dauern, da die JVM anfangs nicht viel Speicher freizugeben hat. Nehmen wir nun an, dass in der JVM alles ordnungsgemäß läuft, und wenden wir uns den Servlets zu.

## SERVLETS

Servlets steuern die J2EE-Transaktion innerhalb der JVM. Ihre optimale Leistung ist von grundlegender Bedeutung für die meisten dynamischen J2EE-Transaktionen. So leiten zum Beispiel viele Websites alle Anfangstransaktionen an ein Anmelde-Servlet weiter. Wenn dieses Anmelde-Servlet langsam reagiert, erlebt jeder Anwender, der sich bei der Website anmeldet, eine lange Reaktionszeit. (Vergessen Sie nicht: Es geht schneller, eine Website der Konkurrenz aufzurufen, als auf die Reaktion einer langsamen Website zu warten.) Ganz offensichtlich ist es wichtig, die Reaktionszeit von Servlets zu überwachen, denn es ist nicht auszuschließen, dass ein Servlet endlos auf eine Backend-Ressource (etwa eine Datenbank) wartet. Einige Java-Programmierer entwerfen Servlets, ohne für Datenbankabfragen der Servlets eine Höchstdauer festzulegen, nach der sie abgebrochen werden. Wenn andere Servlets ebenfalls auf diese Datenbank zugreifen, die nicht reagiert, weist die JVM jedem dieser endlos auf eine Reaktion wartenden Servlets einen Thread zu. Irgendwann stehen der JVM dann vielleicht keine Threads mehr zur Verfügung. Das Ergebnis ist eine Website, die überhaupt nicht mehr reagiert. Durch die Überwachung der Reaktionszeit von Servlets können Sie vermeiden, dass die JVM über zu wenige Threads verfügt.

Zusätzlich zur Überwachung der Reaktionszeit von Servlets sollten Sie die Anzahl der Internet-Sitzungen/-Anfragen je Servlet berücksichtigen. Diese Zahl ist wichtig, denn wenn ein Servlet normalerweise kontinuierlich Anfragen übergibt (zum Beispiel ein Anmelde-Servlet), und die Anzahl der Anfragen plötzlich auf null zurückgeht, könnte dies ein Hinweis auf ein Problem außerhalb der JVM sein. (Möglicherweise stehen dem HTTP-Server keine Listener

mehr zu Verfügung oder es ist ein Netzwerkausfall eingetreten.) Es könnte auch ein Hinweis darauf sein, dass ein Servlet eine Backend-Datenbank abfragt und auf eine Reaktion wartet. Nehmen wir nun an, dass die Servlets wunschgemäß funktionieren, und wenden wir uns den EJB zu.

## ENTERPRISE JAVA BEANS

Auch ohne Enterprise Java Beans können Sie eine robuste Webanwendung erstellen, wenn Sie Ihren Webserver, Servlets und JSP einsetzen. Wenn Ihre Anwendung jedoch Zugriff auf eine zentral definierte Geschäftslogik und/oder auf zahlreiche Backend-Datenbanken benötigt, sind EJB die beste Lösung. EJB fügen zwischen die Ebene der Java-Anwendungen und die Backend-Datenbanken eine Abstraktionsebene ein. Diese Abstraktionsebene sorgt auch für die Koordinierung der Geschäftslogik Ihrer Java-Anwendungen. Sie unterscheidet sich insofern von einem Servlet, als Servlets nur die Programmlogik einer Java-Anwendung koordinieren. EJB müssen ebenso einwandfrei funktionieren wie Servlets, da sie in der J2EE-Anwendungsebene eine wichtige Rolle spielen.

Am besten ist die Gesamtleistungsfähigkeit einer EJB an der Reaktionszeit der einzelnen Methoden (Funktionen) in der EJB zu erkennen. Diese Reaktionszeit ist wichtig, da sie die Reaktionszeit aller EJB widerspiegelt (mit Ausnahme der für die Erzeugung und das Löschen der Bean erforderlichen Zeit). Eine lange Reaktionszeit deutet auf ein schlecht codiertes Servlet hin. Dieses schlecht geschriebene Servlet könnte versuchen, auf eine EJB zuzugreifen (über eine so genannte Entity-Bean) und dabei für eine einzige Datenbankabfrage mehrere Aufrufe absetzen. Derartige Entity-Bean-Verbindungen beeinträchtigen die Leistung erheblich. Ein besser geschriebenes Servlet benötigt für dieselben Datenbankinformationen weniger Aufrufe der EJB (über eine Session-Bean).

EJB werden aus einem Pool in der JVM zugewiesen. Jedes Mal, wenn ein Java-Programm für ein Objekt eine EJB benötigt, wird auf den Bean-Pool zugegriffen. Die durchschnittliche Anzahl der Objekte im Pool sollte überwacht werden. Diese Anzahl ist wichtig, da die Größe des EJB-Pools von zwei Variablen abhängt: der Anzahl und der Größe der Beans. Wenn Ihr Bean-Pool zu groß wird, sinkt die Heap-Größe. Es ist durchaus sinnvoll, für die Größe des EJB-Pools einen Höchstwert zu definieren.

Eine EJB kann auch eine so genannte Message-Driven-Bean aufrufen. Diese Beans dienen der asynchronen Kommunikation mit anderen Ressourcen. Normalerweise rufen sie den Java Message Service (JMS) auf. Sehen wir uns den JMS etwas genauer an.

## JAVA MESSAGE SERVICE

Der JMS ermöglicht einer Java-Anwendung die asynchrone Kommunikation. Er kann mit Servlets oder EJB genutzt werden. In beiden Fällen können Java-Anwendungen über den JMS formlos auf andere Java-Anwendungen zugreifen. Der JMS kann Nachrichten auf verschiedene Art transportieren. Die erste Möglichkeit ist die vom J2EE-Framework bereitgestellte generische JMS-API. Dieser generische Zugriff erfolgt über die nativen Nachrichtenfunktionen des JMS. Die zweite Methode verwendet ebenfalls die JMS-API, für die Übertragung ist jedoch eine nachrichtenorientierte Middleware eines Fremdherstellers zuständig, zum Beispiel WebSphere MQ. Die dritte Möglichkeit, J2EE-Nachrichten zu transportieren, ist nicht Bestandteil des JMS. Dabei wird eine API eines Fremdherstellers aufgerufen und somit direkt, unter Umgehung des JMS, auf den Nachrichtentransport des Fremdherstellers zugegriffen.

Der JMS (und jeder andere Nachrichtendienst) hat auch einige Nachteile:

- > Ist die Nachricht tatsächlich angekommen?
- > Wo ist die Nachricht jetzt?
- > Wann ist die Nachricht angekommen?

Nehmen wir an, Ihre Nachrichten erreichen ihr Ziel über den JMS wie geplant. Nun fragen Sie sich vielleicht, wie all diese J2EE-Ressourcen verwaltet werden. Sehen wir uns daher die Managementschnittstellen von J2EE genauer an.

## J2EE-MANAGEMENTSCHNITTSTELLE

Jede Managementschnittstelle und jedes Managementverfahren greift in gewissem Grad in das System ein. Dieses Eingreifen kann die Betriebseigenschaften einer J2EE-Komponente entscheidend verändern und die Messergebnisse verzerren. Berücksichtigen Sie dies bitte bei jedem der folgenden drei J2EE-Managementverfahren:

- > Java Management Extensions (JMX)
- > Byte-Code Instrumentation
- > Java Profiler Interface (JVMP)

JMX ist das J2EE-Managementverfahren, das am weitesten verbreitet ist und am schnellsten wächst. Es vereinfacht die Verwaltung jeder beliebigen Java-Anwendung (vorausgesetzt, die Anwendung ist JMX-fähig). Dies bedeutet, dass JMX Ihre selbst geschriebenen Java-Anwendungen ebenso verwalten kann wie Anwendungen von Fremdherstellern. Viele beliebte JVM verfügen über eine eigene JMX-Schnittstelle, so dass die JVM selbst über JMX verwaltet werden kann. Unabhängig davon, was Sie mit JMX verwalten, wird das JMX-Management über Management-Beans (MBeans) erledigt. JVM-Hersteller und ISV werden künftig im Bereich der JMX sicher noch wesentlich aktiver werden.

Die Byte-Code-Instrumentation ist ein Verfahren, mit dem Aufrufe dynamisch in Java-Programme eingefügt werden. Diese Aufrufe extrahieren Informationen dynamisch aus den Java-Programmen. Die extrahierten Informationen können zum Beispiel Einzelheiten zu SQL-Anweisungen sein, etwa die Prozessorauslastung je SQL-Anweisung. Ebenso wie PMI verursacht auch die Byte-Code-Instrumentation eine erhebliche Prozessorlast.

Das dritte Managementverfahren ist das JVMP. Das JVMP verursacht von allen J2EE-Managementverfahren die höchste Prozessorlast. Allerdings liefert es sehr feine statistische Daten zu J2EE, zum Beispiel zur CPU-Auslastung durch die JVM und zur Garbage Collection. In Anbetracht der hohen Prozessorlast sollte dieses Managementverfahren nur in Entwicklungsumgebungen eingesetzt werden, nie in der Produktionsumgebung.

### ZUSAMMENFASSUNG UND EMPFEHLUNGEN

Überwachen Sie Ihren Java-Anwendungsserver einige Wochen lang, um die „normalen“ Fluktuationen in der Ressourcenauslastung zu erkennen. Behalten Sie die Speicherauslastung im Auge und achten Sie darauf, dass die Garbage Collection regelmäßig, aber nicht zu häufig stattfindet. Stellen Sie sicher, dass Ihre Servlets nicht zu lange auf Dateien warten. Threads sind eine knappe Ressource; vergewissern Sie sich, dass Ihre Java-Programmierer Threads rechtzeitig freigeben. EJB sind inzwischen weitgehend akzeptiert und viele Unternehmen lernen, wie man sie richtig einsetzt. Denken Sie daran, dass eine EJB genau wie ein Servlet eine Zeitgrenze überschreiten kann, wenn sie auf eine Backend-Ressource wartet. Wenn Sie die Reaktionszeit von EJB und Servlets überwachen, können Sie das „Einfrieren“ Ihrer Website vorhersehen (und hoffentlich verhindern). Setzen Sie mindestens eines der hier besprochenen Java-Managementverfahren ein, zum Beispiel die Java Management Extensions, um detaillierte Managementinformationen zu gewinnen. Welches der verschiedenen Verfahren für Sie am besten geeignet ist, hängt von verschiedenen Faktoren ab, zum Beispiel vom Anwendungsentwurf und der Systemkapazität.

Mit Java können Sie Ihre Anwendungen modular und plattformübergreifend gestalten. Damit können Sie Ihre Hardware- und Entwicklungskosten in Grenzen halten und die Produktivsetzung Ihrer Anwendungen beschleunigen. Wenn diese Anwendungen richtig verwaltet werden, nutzen sie Ressourcen effizienter und sind zuverlässiger. Die größte Herausforderung bei der Verwaltung Ihrer J2EE-Infrastruktur ist jedoch die Sicherstellung der erwarteten Leistungsebenen. BMC Software bietet Lösungen für das Anwendungsmanagement an, die die Überwachung und Optimierung Ihrer J2EE-Anwendungen automatisieren. Die Java-Anwendungsmanagement-Lösungen von BMC Software helfen Ihnen bei der Definition, Messung und Optimierung der Service-Qualität, die Ihre Java-Anwendungen benötigen, und bei der Verwaltung alles dessen, was auf der Ebene der Business-Services wichtig ist.<sup>1</sup>

### WIR HELFEN IHNEN, IHREN WETTBEWERBSVORTEIL ZU WAHREN.

Mit einer umfassenden Service-Suite für das Service-Level-Management unterstützen die BMC Software Professional Services Sie bei der Wahrung des Wettbewerbsvorteils Ihres Unternehmens. Diese Services umfassen Beratung, Installation, Implementierung, Konfiguration und Anpassung. Mit unseren Services und Schulungsangeboten möchten wir dazu beitragen, dass Sie die ununterbrochene Verfügbarkeit wichtiger Geschäftsanwendungen sicherstellen können, Produkte optimal nutzen, das Projektrisiko senken, den Wert der IT für Ihr Geschäft erhöhen und Ihre Operationen verbessern. Weitere Informationen über BMC Software Professional Services finden Sie unter <http://www.bmc.com/profserv>.

1. Weitere Informationen finden Sie unter [www.bmc.com](http://www.bmc.com).



## ÜBER BMC SOFTWARE

BMC Software, Inc. [NYSE:BMC] ist ein führender Anbieter von Enterprise-Management-Lösungen, die Unternehmen in die Lage versetzen, ihre IT-Infrastruktur aus unternehmerischer Sicht zu verwalten. Mit dem Business Service Management deckt BMC Software Unternehmenssysteme, -anwendungen, -datenbanken und das Service-Management ab. BMC Software wurde 1980 gegründet und verfügt heute über Niederlassungen in aller Welt. 2004 betrug der Gesamtumsatz über 1,4 Mrd. USD. Weitere Informationen über BMC Software finden Sie unter [www.bmc.com](http://www.bmc.com).

© 2004 BMC Software, Inc. Unveröffentlichtes Dokument. Alle Rechte vorbehalten. BMC Software, die BMC Software-Logos und alle anderen Namen von Produkten und Dienstleistungen von BMC Software sind eingetragene Warenzeichen oder Marken von BMC Software, Inc. IBM ist ein eingetragenes Warenzeichen der International Business Machines Corporation. DB2 ist ein eingetragenes Warenzeichen der International Business Machines Corporation. Oracle ist ein eingetragenes Warenzeichen und alle Namen von Oracle-Produkten sind eingetragene Warenzeichen oder Marken der Oracle Corporation. Alle anderen Marken sind Eigentum der jeweiligen Unternehmen. 52870 02/05



52870