UNIVERSITÄT LEIPZIG

Mainframe Internet Integration

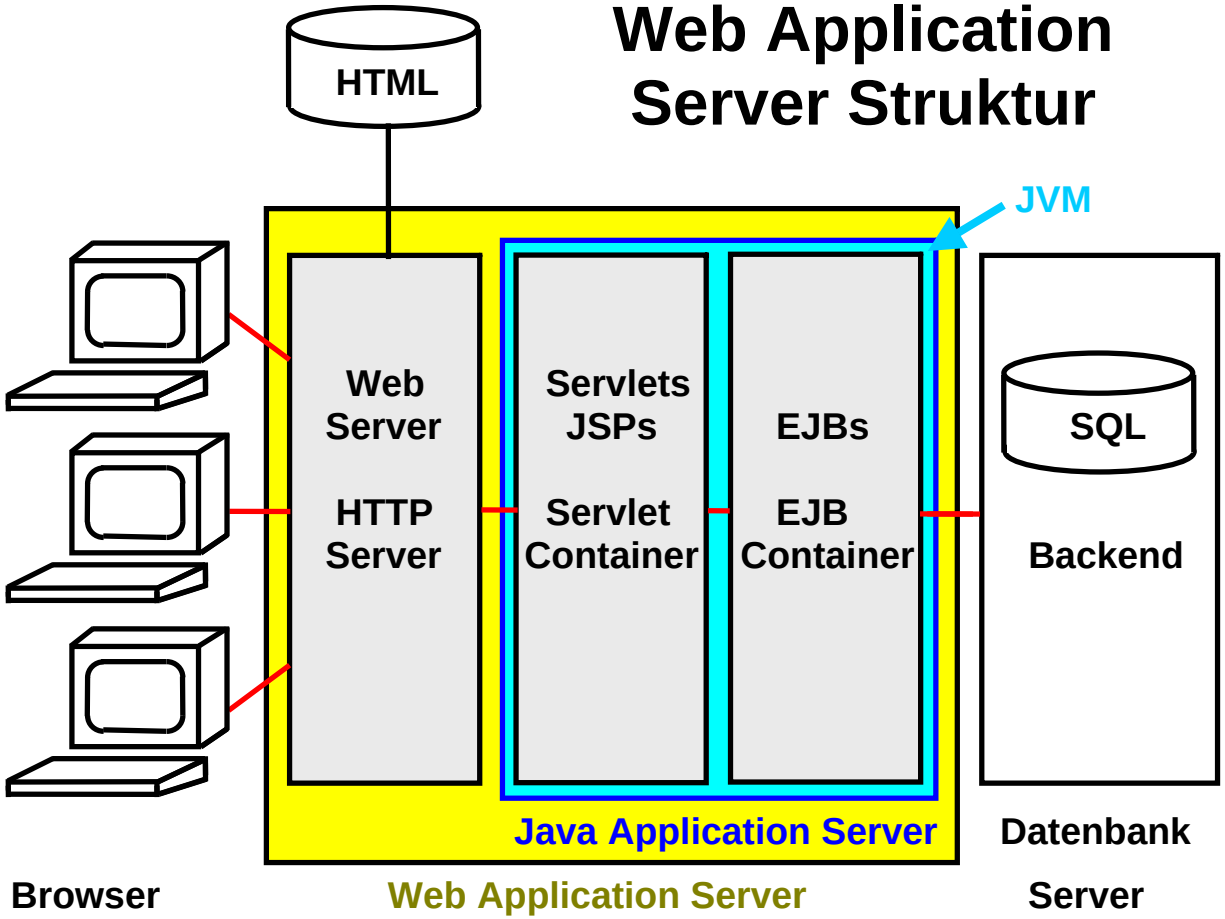
Prof. Dr. Martin Bogdan
Prof. Dr.-Ing. Wilhelm G. Spruth

SS2013

Java Enterprise Edition Teil 4

Schnittstellen

Web Application Server Struktur



Der Web Application Server ist ein Prozess, der normalerweise in seinem eigenen virtuellen Adressenraum läuft. Er besteht aus mehreren Komponenten:

1. Der Web Server ist vielfach Apache.
2. Der Java Application Server unterhält u.a. eine Java Virtuelle Maschine
3. Der Servlet Container (Servlet Engine) ist eine Java Laufzeit Umgebung (runtime component) für die Ausführung von Servlets und Java Server Pages.
4. Der EJB Container ist eine Laufzeit Umgebung für die Ausführung von deployed Enterprise Java Beans.

IBM bezeichnet seinen JEE Server als WebSphere Web Application Server (WAS).

Servlet Container und EJB Container laufen in einer gemeinsamen Java virtuellen Maschine (JVM). Dies reduziert den Kommunikationsaufwand. Servlet Klassen können EJB Klassen direkt aufrufen. Bei getrennten JVMs erfolgt die entsprechende Kommunikation mit Hilfe des RMI oder RMI/IIOP Protokolls.

Der Web Application Server enthält weitere Elemente für Administration und Datenbankzugriff.

HTTP Server

Apache ist der am weitesten verbreitete http Server. Apache läuft unter AIX, HP-UX, Linux, Solaris, Window, und z/OS. Apache wird von einer Open Community von Entwicklern unter der Leitung der Apache Software Foundation kontinuierlich verbessert.

Der IBM HTTP Server basiert auf dem Apache http Server. Das Download ist kostenlos. Der IBM HTTP Server ist ebenfalls ein Teil der IBM WebSphere Application Server Distribution Package.

Neben Apache existieren zahlreiche weitere http Server. Am bekanntesten ist der Microsoft Internet Information Server (IIS), vorgesehen für eine Benutzung mit Microsoft Windows. Dies ist nach Apache der zweitwichtigste http Server.

Der IBM Lotus Domino Server ist ein spezialisiertes Software Product mit einer eigenen Datenbank (Notes Storage Facility). Der Lotus Domino HTTP Server wird gelegentlich auf Mainframes eingesetzt.

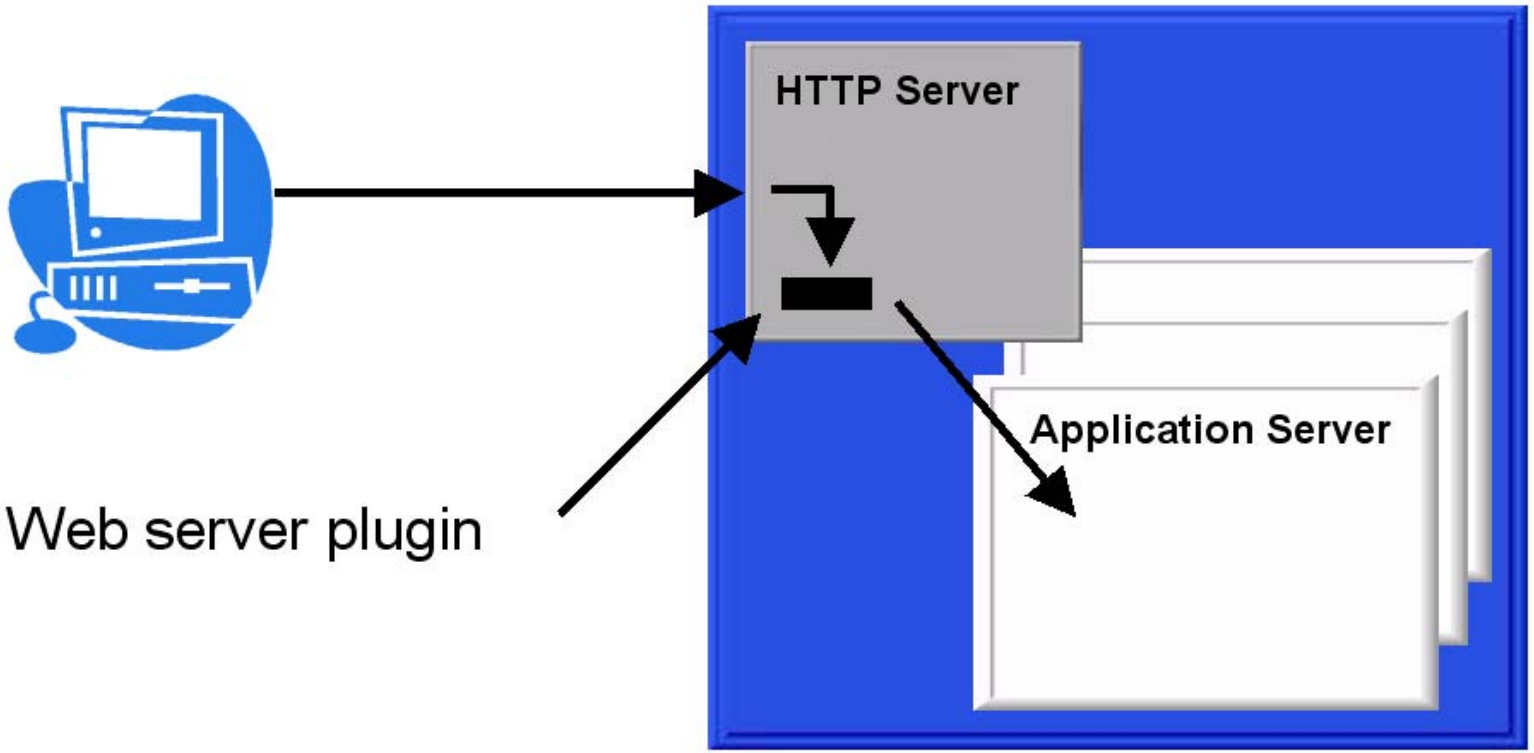
Daneben existieren zahlreiche weitere HTTP Server Produkte.

Zweideutige Bezeichnung

In der Vergangenheit war ein Web Server immer ein Server, der eine URL auswerten und statische HTML Seiten an einen Klienten (Browser) zurückliefern konnte. Bei der Einführung von CGI und Servlets erfolgte eine Ergänzung (Plug-in), um beim Parsen von CGI oder Servlet Tags das entsprechende CGI oder Servlet Programm aufrufen zu können.

Später erfolgte eine Begriffsänderung (Begriffsverwirrung). Man begann komplexe WWW Anwendung unter Nutzung von Servlets, JSPs und EJBs zu erzeugen, sowie viele unterschiedliche derartige Anwendungen auf dem JEE Server zur Verfügung zu stellen. Zu einer vollständigen Anwendung (z.B. Banküberweisung, Hotelreservierung, Bestellung beim Otto Versand) besteht diese neben Servlets, JSPs und EJBs zusätzlich auch aus einer Reihe von statischen HTML Seiten. Man fasst diese anwendungsspezifischen statischen HTML Seiten mit den Servlets und JSPs zu einer Einheit zusammen, und nennt diese Einheit Web Server, Web Engine oder Web Container. Den bisherigen Web Server bezeichnet man zur Unterscheidung als http Server. Der http Server verwaltet alle die HTML Seiten, die nicht einer Anwendung spezifisch zugeordnet werden können.

Leider ist diese Zweideutigkeit bis heute geblieben. Es ist deshalb sinnvoll einen Server für statische HTML Seiten grundsätzlich als http Server zu bezeichnen.



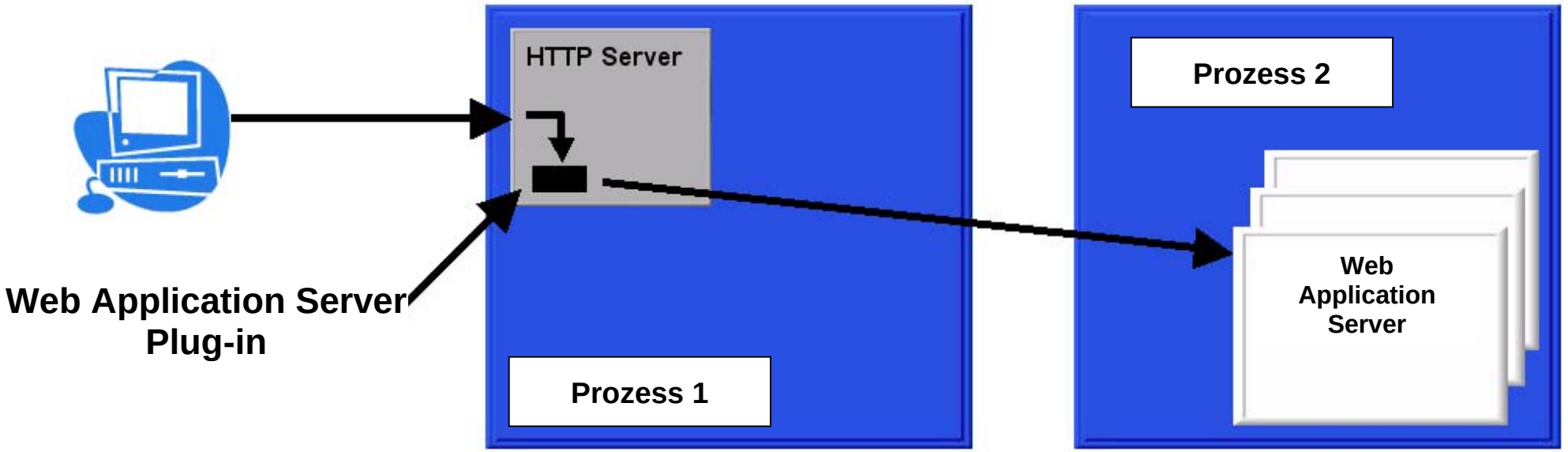
Es besteht der Wunsch der Benutzer von JEE Servern wie Weblogic, WebSphere Web Application Server (WAS) und Netweaver, mit unterschiedlichen http Server Produkten zusammen arbeiten zu können. Deswegen enthält eine JEE Server Distribution mehrere Interface Komponenten, die eine Verbindung mit unterschiedlichen http Servern ermöglichen. Diese Interface Komponenten werden als „Plug-in“ bezeichnet.

Die WebSphere Web Application Server Distribution enthält z.B. Plug-in’s für Apache, den IBM Http Server, IIS und den Lotus Domino http Server. Bei der Installation wird das Plug-in in den http Server integriert.

Die ursprüngliche WebSphere Application Server Standard Edition für OS/390 (verfügbar in 1999) lief in dem gleichen Adressenraum wie der HTTP-Server.

Der Web Application Server war ursprünglich als Erweiterung zu einem http-Server gedacht, bestehend aus 2 Hauptkomponenten:

- **Ein Plug-in für den Web-Server (HTTP-Server), welcher die Anforderung an den tatsächlichen Application Server weiterreicht, und**
- **Der Application Server selbst**



Heute läuft ein Web Application Server als eigener Prozess, in der Regel auf der gleichen Maschine wie der HTTP-Server (Web-Server). Es könnte auch auf einem anderen Rechner laufen. Es können mehrere Application Server Kopien als unabhängige Prozesse konfiguriert werden.

Das Webserver Plug-in ist ein Teil der Web Application Server Software-Package, wird aber in dem HTTP-Server-Adressraum installiert. Es kann mit dem Application Server über mehrere Protokolle kommunizieren, abhängig von dem Application Server Hersteller. IBM WebSphere WAS, einschließlich der z/OS-Version verwendet HTTP und HTTPS.

Dies ist der Grund dafür:

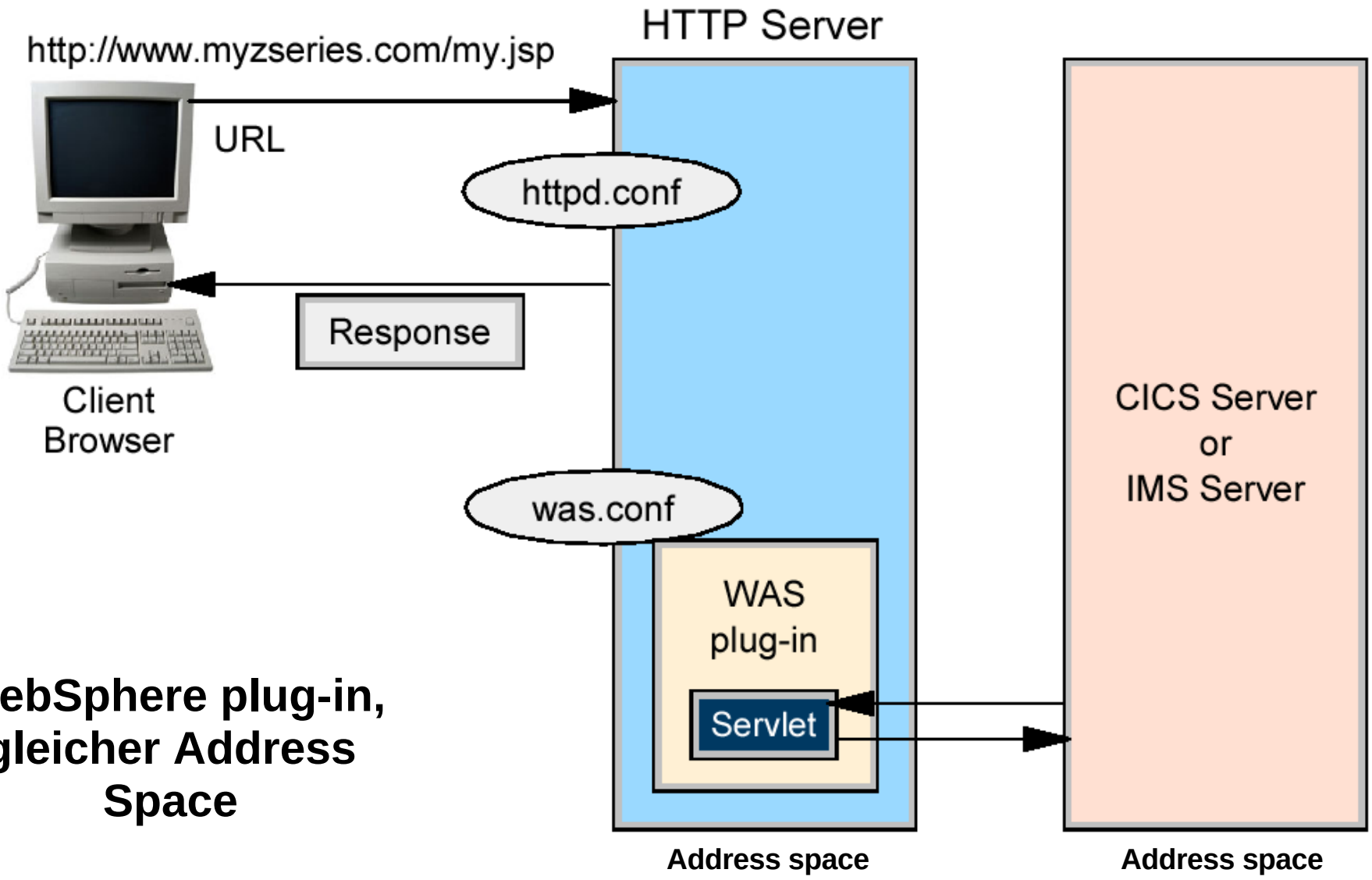
Ein WebSphere WAS Software Package enthält den IBM HTTP Server-Code, der als Standard-Option installiert ist. Sie kann jedoch konfiguriert werden, einen anderen Web Server zu benutzen,, z. B. Microsoft IIS. Damit IIS auf den WebSphere WAS zuzugreifen kann, muss der mit WebSphere mitgelieferte Plug-in in dem IIS-Adresse Raum installiert werden.

Plug-in code

Wenn Anforderungen in dem HTTP Server eintreffen, entscheidet die HTTP-Server-Konfigurationsdatei (httpd.conf), ob die Anfrage an den Plug-in code weitergereicht werden soll.

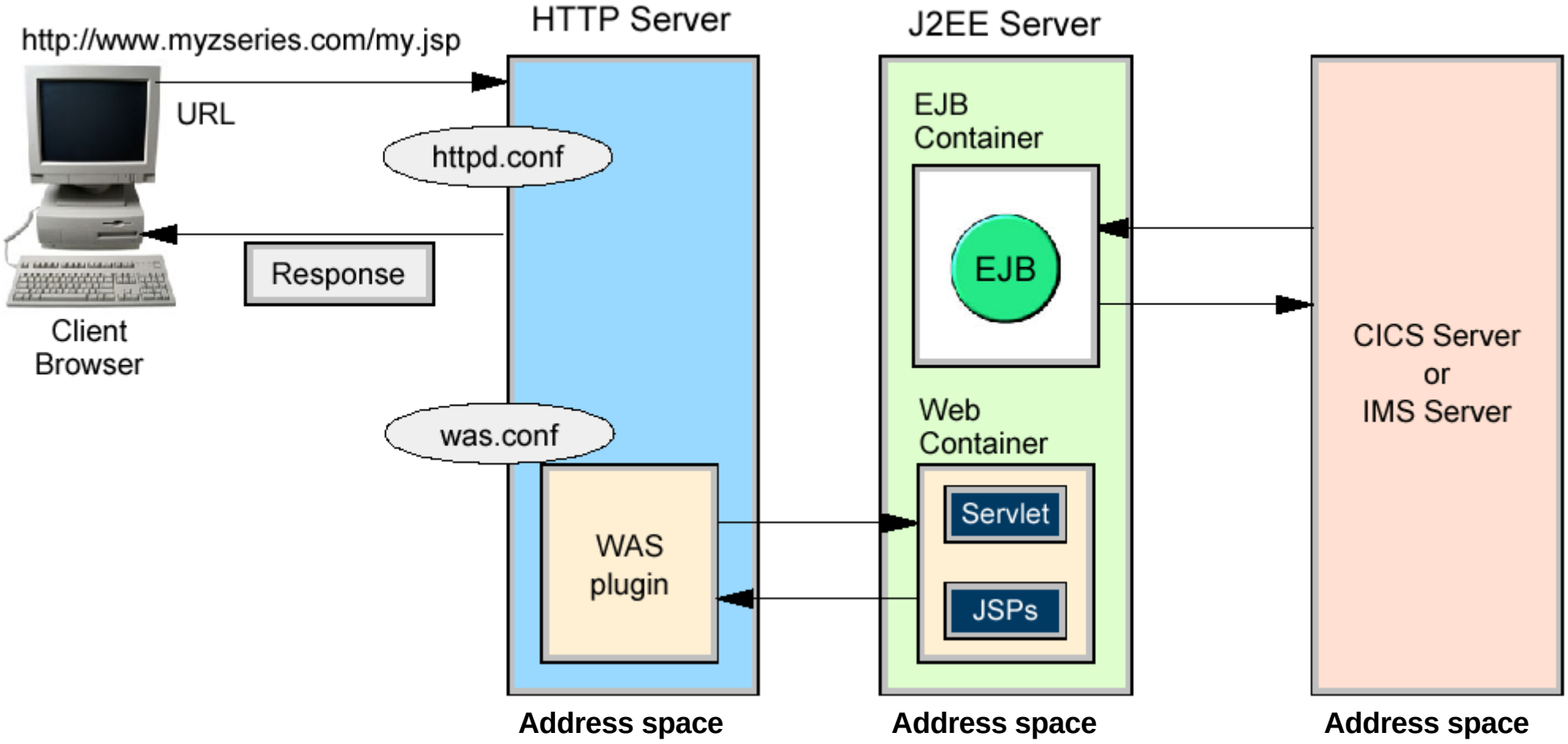
Das Plug-in enthält eine eigene Konfigurationsdatei. Diese entscheidet, an welchen von potentiell mehreren Application Servern die Anforderung weiter gereicht wird.

Die folgenden Abbildungen zeigen zwei mögliche Web Application Server Konfigurationen.



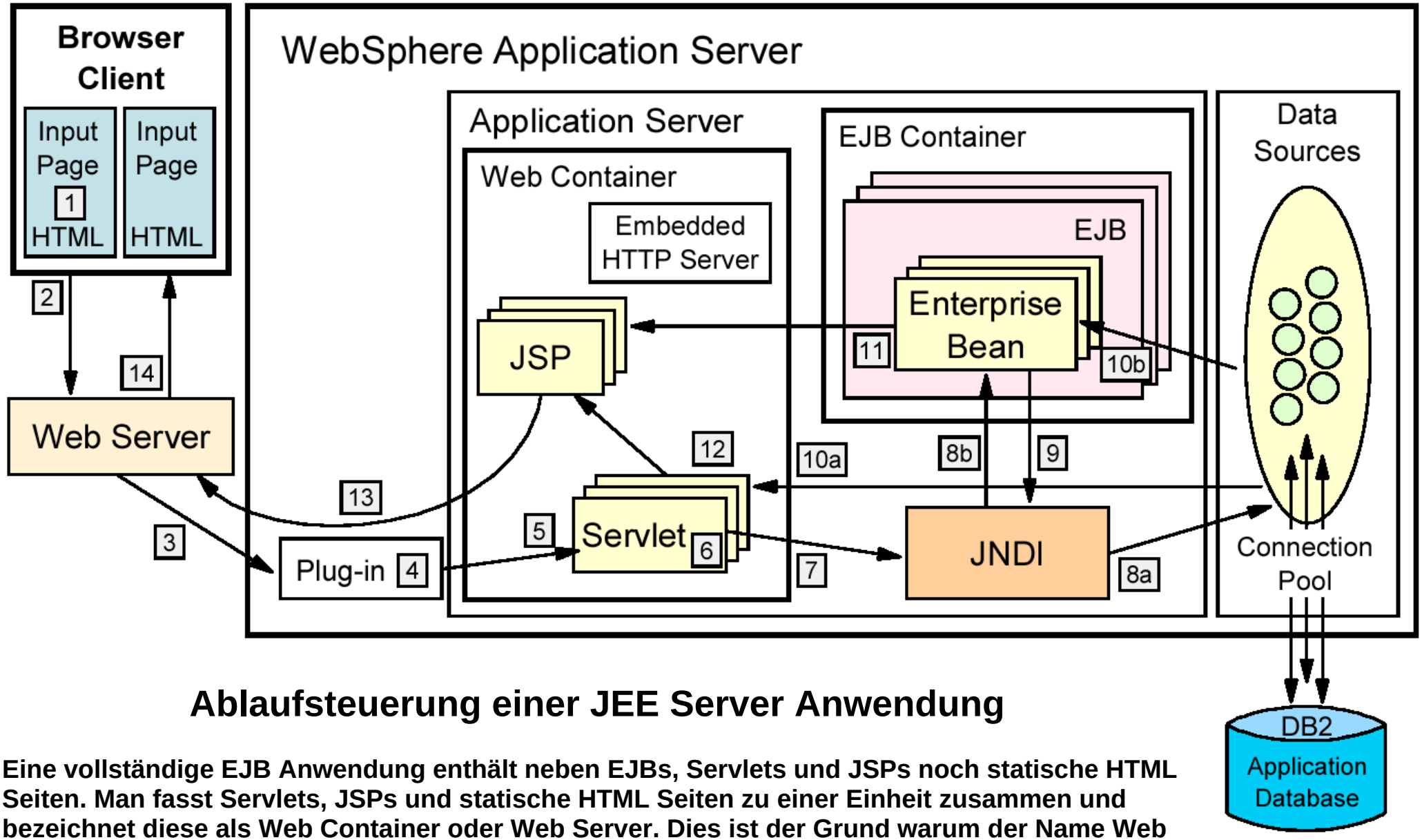
WebSphere plug-in, gleicher Address Space

Diese Abbildung zeigt eine einfache Konfiguration, in der kein JEE-Server benötigt wird. Der Servlet Container befindet sich in dem gleichen Adressenraum wie der http Server. Das Servlet kann CICS, IMS oder DB2 über JDBC aufrufen. Allerdings wird eine Kodierung von Geschäftslogik innerhalb des Servlets nicht empfohlen.



Getrennter JEE Server mit Web Container und EJB Container

Sie können auch das WebSphere Plug-in benutzen, um Servlets remote in einem „Web-Container“ auszuführen. Dies ermöglicht es, Servlets und EJBs in dem gleichen Adressraum auszuführen, so dass keine Remote EJB Aufrufe (über das RMI-Protokoll) benötigt werden.

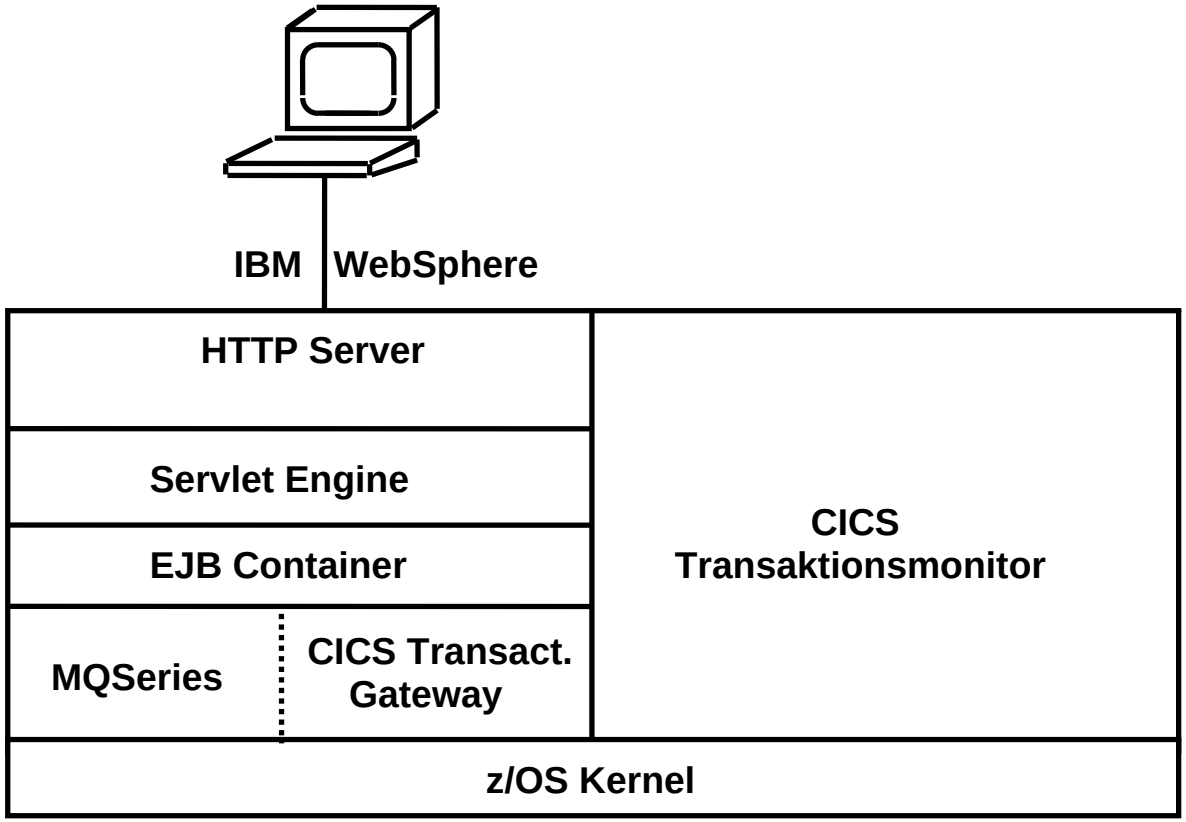


Ablaufsteuerung einer JEE Server Anwendung

Eine vollständige EJB Anwendung enthält neben EJBs, Servlets und JSPs noch statische HTML Seiten. Man fasst Servlets, JSPs und statische HTML Seiten zu einer Einheit zusammen und bezeichnet diese als Web Container oder Web Server. Dies ist der Grund warum der Name Web Server doppelt belegt ist, und man Apache z.B. als http Server bezeichnen sollte.

Der typische Ablauf der Ausführung einer JEE Server Anwendung ist wie folgt:

1. **Ein Web-Client ruft eine URL in seinem Browser auf (Input Page).**
2. **Die Anfrage wird über das Internet an den Web Server (http Server) geroutet.**
3. **Der Web-Server leitet die Anfrage an das Web Server Plug-in weiter.**
4. **Das Web Server Plug-in prüft die URL, und wählt einen Server aus, um die Anfrage zu bearbeiten.**
5. **Ein Stream wird erzeugt. Ein Stream ist eine Verbindung mit einem Web-Container. Es ist möglich, eine Verbindung (Stream) über eine Anzahl von Anfragen aufrecht zu erhalten. Der Web-Container erhält die Anfrage und reicht sie an das richtige Servlet weiter.**
6. **Wenn die Servlet-Klasse nicht geladen ist, lädt der dynamische Klassenlader das Servlet (Servlet init() Methode), und ruft dann die doGet() oder doPost() Methode auf.**
7. **JNDI wird benutzt, um die Lokation entweder der Data Source (Datenbank, VSAM) oder der EJBs zu finden, die von dem Servlet benötigt werden.**
8. **Je nachdem, ob eine Data Source angegeben ist oder eine EJB angefordert wird, directet JNDI das Servlet zu:**
   - **der entsprechende Datenbank. Hierfür wird eine bereits existierende Verbindung von dem Connection Pool zugeordnet.**
   - **dem entsprechenden EJB-Container, welcher dann die EJB instanziiert.**
9. **Wenn die EJB eine SQL-Transaktion beinhaltet, benutzt sie JNDI zum Nachschlagen der Data Source.**
10. **Die SQL-Anweisung wird ausgeführt, und die aufgerufenen Daten werden entweder zu dem Servlet oder der EJB zurück gesendet.**
11. **Data Beans werden erstellt und im Falle einer EJB an eine JSP weiter gereicht.**
12. **Das Servlet sendet Daten an die JSP.**
13. **Die JSP generiert eine HTML Seite, die über das Plug-in an den Web-Server weiter gereicht wird.**
14. **Die Web-Server sendet die Output-Seite (Ausgabe HTML) an den Browser. Diese wird dort zur Input Page für den nächsten Schritt.**



Web Application Server als Tranaktionsmonitor Front End

Beispiel: WebSphere als Front End für den CICS Transaktionsmonitor

Obwohl EJBs Transaktionseigenschaften haben, und die gesamte Business Logik implementieren könnten, setzt man in der großen Mehrzahl der Fälle einen traditionellen Transaktionsmonitor wie CICS für den Kern der Business Logik ein.

JBoss und Geronimo Web Application Server

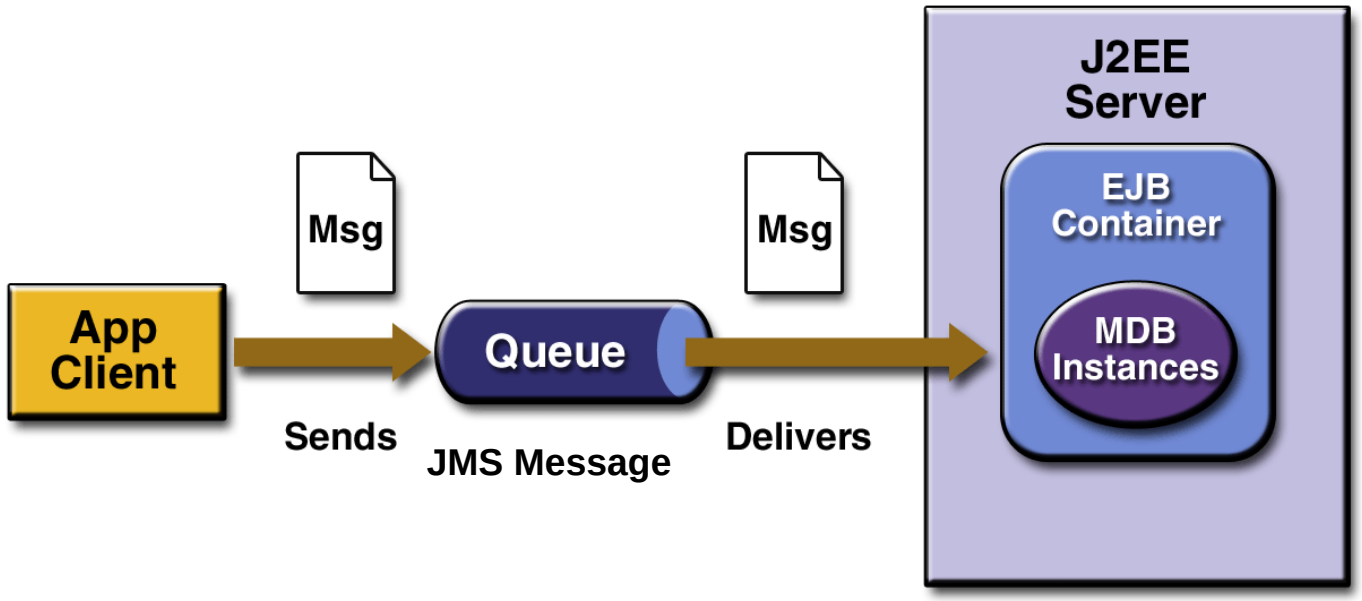
JBoss und Geronimo sind Opensource-Implementierungen eines Web Application Servers.

Die eigenständige Servlet-Laufzeitumgebung Tomcat (ebenfalls Open Source) ist als Webcontainer (Servlet Container) standardmäßig in JBoss und Geronimo integriert.

JBoss und Geronimo sprechen ihren Transaktionsmanager (bis auf den Import und Export des Transaktionskontextes) allein über die JTS-Schnittstellen an, sodass sich verhältnismäßig einfach beliebige JTS-konforme Transaktionsmanager integrieren lassen.

Jboss und Geronimo haben nicht den Reifezustand und Funktionsumfang von kommerziellen Web Application Servern wie Oracle WebLogic oder IBM WebSphere. JBoss besteht aus knapp 4.300 Klassen und belegt installiert 50 Mbyte auf dem Festplattenspeicher, Websphere besteht aus über 20.000 Klassen und belegt installiert 460 MByte.

GlassFish und Geronimo (Apache Foundation) sind weitere Open Source Software Produkte.



Message Driven Bean

Der JEE Java Message Service (JMS) ermöglicht eine Nachrichtenübermittlung durch asynchrone Methodenaufrufe

Eine Message Driven Bean (MDB) ist eine spezielle Art einer Enterprise Java Bean (EJB). Eine Message Driven Bean ist ein potentieller Empfänger eines asynchronen Methodenaufrufes. Dieser Methodenaufruf kann von einer beliebigen anderen Java Klasse ausgehen. Die sendende Klasse wartet nicht auf eine Antwort.

Im Gegensatz zu Session Beans und Entitiy Beans erfolgt der Zugriff auf eine MDB nicht über eine Schnittstelle (Interface).

Alle Instanzen eines MDB – Typs sind gleich, da sie nicht direkt für den Client sichtbar sind und keine Zustände annehmen. Daraus resultiert die Möglichkeit für den EJB Container, Pools aus MDB – Instanzen zu bilden, um Skalierbarkeit zu erzeugen.

Message based Queuing (MBQ)

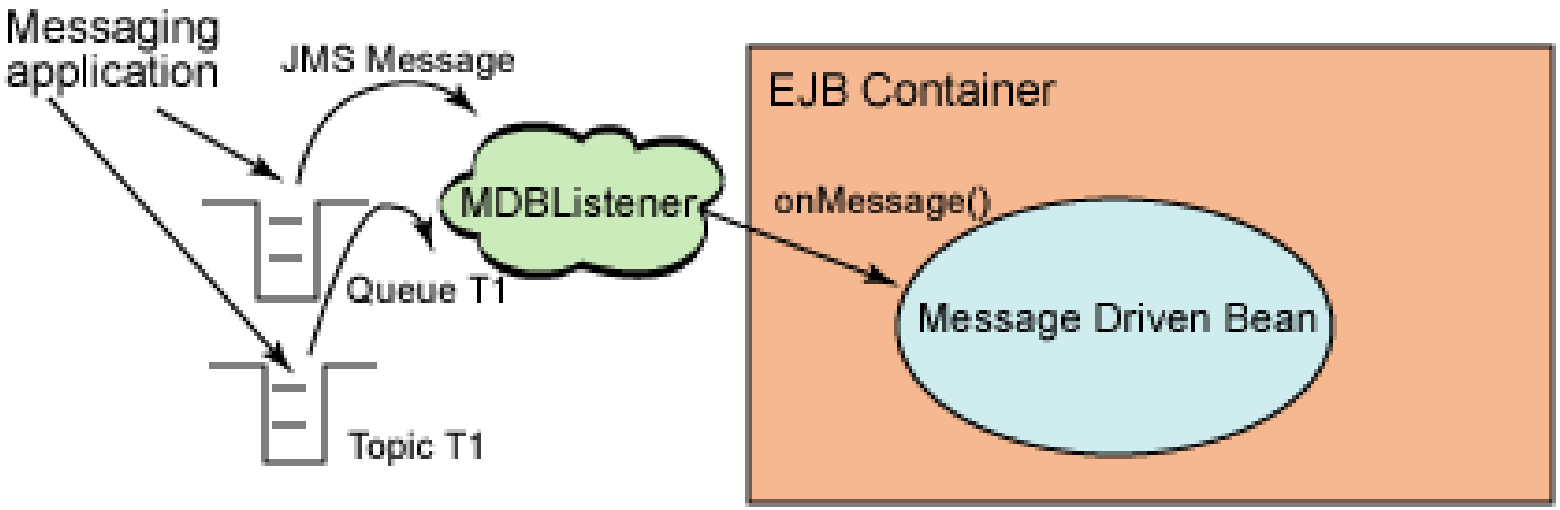
MBQ ist ein Verfahren zur Program to Program Kommunikation. Es kann verwendet werden, um einen asynchronen RPC implementieren. Programme werden in die Lage versetzt, ohne eine direkte Verbindung Informationen zu senden oder zu empfangen. Programme kommunizieren durch das Hinterlegen von Nachrichten in Message-Queues, und Abrufen von Nachrichten von Message-Queues.

Der Service, der einen MBQ Nachricht empfängt kann als reguläre Java-Klasse implementiert werden. Alternativ kann der Service als EJB, einer Message Drive Bean (MDB), implementiert werden.

Der Websphere Web Application Server (WAS) benutzt das MQSeries Produkt, um die MDB Unterstützung zu implementieren. MQSeries existiert in 2 Versionen:

- **Die reguläre MQSeries Version, die alle Sprachen unterstützt. Diese Installation erfordert keinen Websphere Application Server, wird aber dennoch heute als WebSphere MQ bezeichnet.**
- **Eine spezielle MQseries Version, die MDBs unterstützt und die in WebSphere integriert ist Diese unterhält Queues für die MDBs, sowie eine Listener Komponente, welche eine spezifische MDB benachrichtigt, dass in ihrer Queue eine Nachricht eingetroffen ist, die von der onMessage() Methode der MDB empfangen werden kann.**

Beide Versionen verwenden den gleichen Code. IBM hat daher beschlossen, beide Versionen in WebSphereMQ umzubenennen. Die reguläre MQSeries Version kann (und häufig wird) ohne WebSphere verwendet werden.



Message listener service

MDBs hören (listen) ein bestimmtes JMS-Ziel ab. Dies kann zum Beispiel eine Queue sein. Die Message-Client sendet Nachrichten an die jeweilige Queue, auf die die Message Driven Bean zuhört.

Ein MDB-Listener ist ein Service eines WebSphere Application Servers. Der MDB-Listener bewirkt, dass eine MDB (Message-Driven Bean) informiert wird, dass eine Nachricht an einem bestimmten Ziel angekommen ist . Wenn die Queue eine Nachricht empfängt, ruft der EJB-Container die onMessage ()-Methode der Message-Driven Bean auf. Die onMessage () Methode kann auch als einzige Schnittstelle zu einem MDB betrachtet werden.

Ein Listener-Port ist eine Komponente des Applikationsservers, der ein JMS-Ziel und das Eintreffen von Nachrichten überwacht. MDBs sind an einen bestimmten Listener-Port gebunden. Wenn eine Nachricht an dem Listener-Port ankommt, wird sie an die entsprechende MDB (Message-Driven Bean) weitergeleitet..

In EJB 3.0 wird der MDB-Listener durch die „Aktivierungsspezifikationen“ ersetzt.

Message Driven Beans implementieren eine Enterprise Java Bean Variante einer Message Oriented Middleware wie z.B. MQSeries (siehe Vorlesungsscript Teil 10 aus dem WS 2010/11). Der WebSphere Application Server verwendet MQSeries Software Komponenten für die Implementierung der Message Driven Beans Infrastruktur.

Ein wesentlicher Unterschied zwischen den beiden MQ Varianten besteht darin, dass MDBs an Stelle des MQ Channels und der MQI API Enterprise Java Bean spezifische Konstrukte benutzen. Siehe hierzu unser EJB Tutorial 03, sowie das Thema WebSphere MQ, Teil 1, aus dem Vorlesungsscript Einführung in z/OS.