

Web-Datenbank Anbindung

mit Java Servlets basierend
auf Apache / JServ / Oracle8i

Autoren: Christoph Gächter / Martin Zahn
Copyright © 1999 Akadia AG
All rights reserved

Akadia AG
Information Technology
Zieglerstrasse 34
CH-3007 Bern
Tel 031 385 30 30
Fax 031 385 30 34
E-Mail info@akadia.com
Web <http://www.akadia.com>

Einführung

Im «Internet-Zeitalter» wird die Abfrage und Darstellung von Informationen aus Datenbanken immer wichtiger. Dabei führen verschiedene Wege von der Datenquelle zum Anwender. Wir haben die drei serverbasierten Varianten **CGI/DBI/Perl**, **Apache/JServ** und **Oracle Application Server** untersucht und in Kundenprojekten erfolgreich eingesetzt.

Ein betrachten folgenden Einsatz als geeignet:

- **CGI/DBI/Perl** für kleinere Projekte. DBI ist eine neutrale Datenbank Schnittstelle für Perl5. Datenbank Driver für Oracle und viele weitere Datenbanken sind in Form von Perl Modulen vorhanden. Weitere Angaben finden Sie unter:
<http://www.symbolstone.org/technology/perl/DBI/index.html>
http://www.cpan.org/modules/by-authors/Tim_Bunce/
- **Apache/JServ** für mittlere Projekte mit komplexeren Abfragen und Objektmodellen. In unserem Beitrag wollen wir diese Variante näher vorstellen.
- **Oracle Application Server** für umfangreiche Projekte. Weitere Angaben zum Oracle Application Server finden Sie unter:
<http://technet.oracle.com/>

Apache / JServ

Apache / JServ basiert auf Java Servlets und bietet ein Maximum an Plattformunabhängigkeit. Wir haben den Einsatz auf NT4 und RedHat Linux 6.0 getestet. Portierungen auf andere Unix Systeme wie SUN Solaris 2.6 sind von anderen Institutionen getestet worden.

Apache / JServ unterstützt die Entwicklung von echten Multi-Tier Applikationen. Mit Apache steht ein sehr robuster und weit verbreiteter Web-Server zur Verfügung. Zu betonen ist weiterhin, dass alle Komponenten im Web zur Verfügung stehen und heruntergeladen werden können (Siehe Download Site in untenstehender Tabelle). Eine oft genannte Kritik, dass Java «langsam» ist, trifft beim Einsatz von Java als Servlets weniger in den Vordergrund, da Servlets nach dem Laden im Speicher gehalten werden. Als besonderes Feature betrachten wir die Möglichkeit, zeitintensive Operationen wie beispielsweise das Öffnen einer JDBC Datenbank Verbindung in einer Java Klasse zu kapseln. Wird die Web Applikation gestartet, so wird diese Klasse in den Speicher geladen und die DB Verbindung bleibt für weitere HTTP Requests geöffnet, dadurch erzielt man grosse Performance Gewinne.

Benötigte Komponenten

Folgende Komponenten werden für Apache / JServ¹ benötigt:

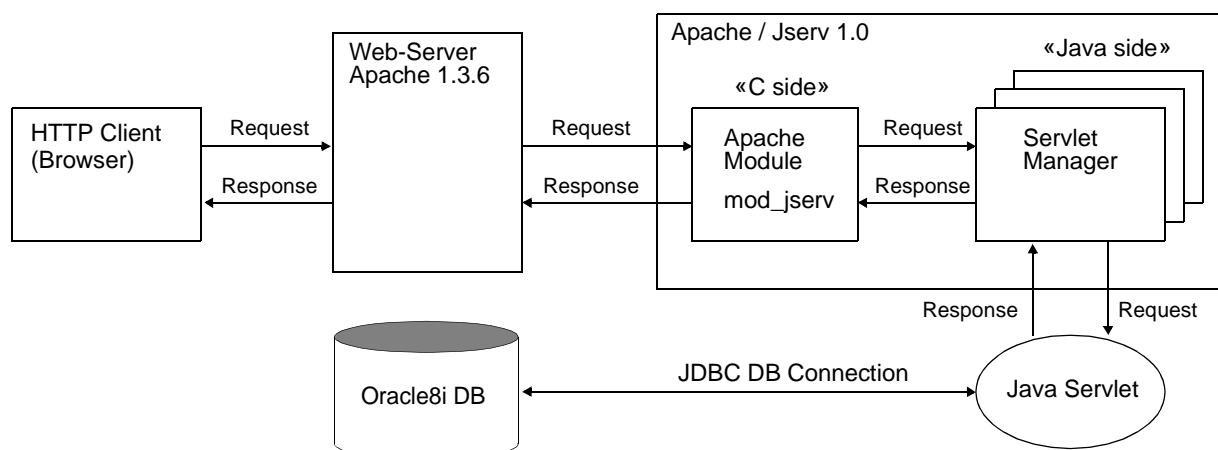
Komponente ^a	Version	Download Site
Java Runtime Umgebung (Java Virtual Machine)	JRE 1.1.6	ftp://iodynamics.com/pub/mirror/linux-jdk/JDK-1.1.6/i386/glibc/v5
Java Development Kit	JDK 1.1.6	ftp://iodynamics.com/pub/mirror/linux-jdk/JDK-1.1.6/i386/glibc/v5
Java Servlet Development Kit	JSDK 2.0	http://java.sun.com/products/servlet/index.html
Apache JServ Module	1.0	http://java.apache.org/jserv/dist
Apache Web Server	1.3.6	http://www.apache.org/dist/

- a. Wir offerieren Ihnen einen Installations- und Konfigurationssupport für NT4 und Unix Systeme.

1. Wir beschränken uns hier auf die Installation für RedHat Linux 6.0, Stand Mitte August 1999

Technologie¹

Als Web Server wird Apache 1.3.6 eingesetzt. Dieser unterstützt auf ideale Art und Weise Apache / JServ. JServ setzt sich aus zwei Hauptkomponenten zusammen, dem eigentlichen Servlet Engine Server und einem Web Server Modul, welches die Kommunikation mit dem Web Server erlaubt und entsprechende Anfragen verarbeitet.

**Ablauf**

Der HTTP Client sendet mittels URL einen Request an den Web Server (z.B. <http://www.akadia.com/abc/JdbcTest>). Der Web Server analysiert den URL. Ist darin der Name einer Servlet Zone enthalten, wird der Request an das Modul Apache Module *mod_jserv* weitergeleitet. In diesem Modul wird die Verbindung zum entsprechenden Servlet Manager aufgebaut in der Form: `<protocol>://<host>:<port>` und autorisiert.

Der Servlet Manager instanziert ein Objekt der Klasse *JServConnection*, welches den Request weiter verarbeitet, dabei das gewünschte Servlet ausführt und die Antwort an *mod_jserv* zurückgibt. Danach wird die Verbindung geschlossen, der Thread sowie das Objekt selber werden im Memory freigegeben.

Servlet Zone

Eine Servlet Zone definiert eine Menge von zusammengehörenden Servlets bzw. Packages (Servlet Bibliotheken), welche in demselben Repository abgelegt sind und über dieselben Konfigurationsparameter verfügen. Jede Servlet Zone wird über einen eigenen Servlet Manager verwaltet.

Java Servlets

Java Servlets sind Module, welche in einem Request / Response (Anfrage / Antwort) orientierten System ausgeführt werden, wie z.B. in einem Java-fähigen Web Server wie Apache. Im Servlet können Daten (z.B. aus einer HTML-Form) entgegengenommen und anhand der definierten Business Logic verarbeitet werden. Das Servlet API (Application Programming Interface), welches zur Implementation von Servlets verwendet wird, ist soweit abstrahiert, dass keine bestimmten Annahmen über das Laden des Servlets oder dessen Run Time Environment vorausgesetzt werden: Damit steht der Einbettung desselben Servlets in unterschiedlichen Web Servern nichts im Weg.

1. Sie können diese Konfiguration direkt auf unserer Website testen:
<http://www.akadia.com/abc/HelloAkadia> (Einfaches Java Servlet ohne DB Zugriff)
<http://www.akadia.com/abc/JdbcTest> (Komplexeres Java Servlet mit Oracle 8i DB Zugriff)

JDBC

Innerhalb des Servlets können Daten von der Datenbank abgefragt oder auf diese geschrieben werden. Dazu kann einer der drei JDBC-Treiber (Thin, OCI7, OCI8) verwendet werden. Die Datenbank-Verbindung kann dabei temporär auf- und wieder abgebaut oder permanent gehalten werden. Dazu sind Kenntnisse über den Lebenszyklus eines Servlets notwendig:

1. Das Servlet wird geladen (Methode `init()`). Dies erfolgt entweder beim erstmaligen Aufruf der Servlets oder aber beim Start von Apache / JServ. Servlet spezifische oder globale Parameter können eingelesen und verarbeitet werden.
2. Nachdem das Servlet geladen und initialisiert worden ist, kann die Methode `service()` ausgeführt werden. Bei HTTP Servlets unterstützt diese Methode die HTTP-Interaktionen GET (Methode `doGet()`), POST (`doPost()`), PUT (`doPut()`) und DELETE (`doDelete()`).
3. Das Servlet bleibt solange geladen, bis es aufgrund eines System Request entfernt wird. Dabei wird die Methode `destroy()` ausgeführt, um allfällige Aufräumarbeiten auszuführen.

Installation

In diesem Abschnitt beschreiben wir kurz die Installation der einzelnen Komponenten Apache Web Server, JDK, JRE, JSDK und JServ.

Apache

Apache 1.3.6 ist im Umfang von RedHat Linux 6.0 bereits vorhanden und muss nicht extra installiert werden. Muss Apache nachträglich installiert werden, so ist darauf zu achten, dass die Apache Module dynamisch geladen werden können (DSO-Support). Weiterführende Hinweise finden Sie auch auf unserer Website.

JRE, JDK und JSDK

JRE ist die Runtime Java Umgebung, JDK ist das Java Development Kit und JSDK¹ wird zur Entwicklung und zum Testen von Servlets verwendet. Alle Sourcen stehen im Web zur Verfügung (siehe «Benötigte Komponenten» auf Seite 1). Man beachte, dass für RedHat Linux 6.0 unbedingt die `glibc` und nicht die älteren `libc5` Sourcen heruntergeladen werden. Die TAR Archive werden in einem beliebigen Directory ausgepackt, in das Zieldirectory verschoben und via Symbolic Link dem System über die PATH Umgebungsvariable bekannt gemacht.

JServ

Bei der Installation von JServ kann der vorhandene Apache 1.3.6 Web Server von RedHat Linux 6.0 verwendet werden.

Der Servlet Server wird erstellt, indem unter Linux das File `apache-JServ-1.0b3.tar.gz` expandiert wird. Im Hauptverzeichnis von Apache JServ können danach folgende Befehle ausgeführt werden:

```
./configure --prefix=/usr/local/jserv
make
make install
```

Als Resultat sind unter `/usr/local/jserv` folgende Verzeichnisse und Dateien erstellt worden:

```
/usr/local/jserv
/usr/local/jserv/doc
/usr/local/jserv/bin
/usr/local/jserv/lib/Apache-JServ.jar
```

1. Es wird dringend empfohlen, exakt die Version JSDK 2.0 im Zusammenhang mit JServ 1.0 zu verwenden.

Das Web Server Modul wird erstellt, indem die C-Source-Files in das Extra-Module-Verzeichnis von Apache kopiert und dort compiliert und gelinkt werden:

```
./configure \
--prefix=/usr/local/apache \
--add-module=src/modules/extra/mod_jserv.c \
--enable-shared=jserv \
--enable-module=jserv
make
make install
```

Das Resultat ist unter `/usr/local/apache` zu finden. In diesem Fall interessiert jedoch nur das Shared Object `mod_jserv.so`, das zu den bereits existierenden Modulen kopiert werden kann.

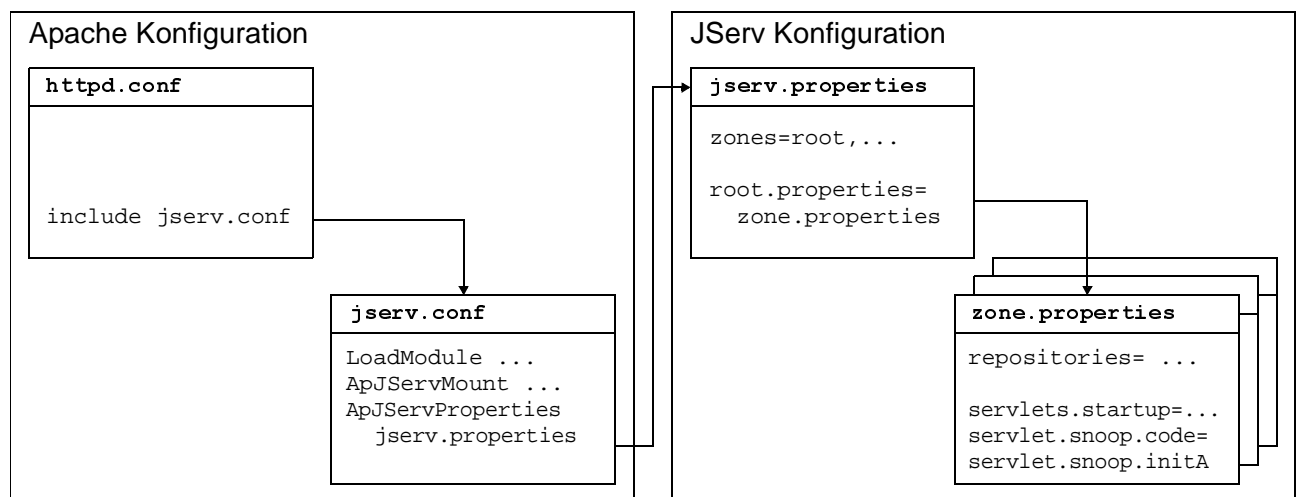
Konfiguration

Es wird empfohlen, folgende Konfigurationen nach der Installation vorzunehmen:

```
export JSDK_HOME=/usr/local/jsdk
export PATH=$PATH:$JSDK_HOME/bin
export CLASSPATH=$CLASSPATH:$JSDK_HOME/lib/jsdk.jar
```

Damit stehen dem Environment einerseits der *Servlet Runner*, andererseits die Java Extensions *jsdk.jar* zur Verfügung.

Apache und JServ werden so konfiguriert, dass sie einerseits miteinander kommunizieren können, andererseits die Umgebung zur Ausführung von Java Code gesetzt ist. Vom Verzeichnis *ServerRoot* ausgehend werden Unterverzeichnisse bzw. Links zu den einzelnen Dateien definiert.



Es können mehrere Servlet Zonen (auf verschiedenen Servern) definiert und konfiguriert werden. Jede Servlet Zone verfügt über ein eigenes Konfigurations-File. Beim Starten von AWS werden die Servlet Zonen in den Web Server File Space „gemountet“.

Weiterführende Informationen

<http://java.apache.org>
<http://java.apache.org/faq>
<http://java.sun.com>
<http://java.sun.com/products/servlet>
<http://www.apache.org>
<http://www.blackdown.org>