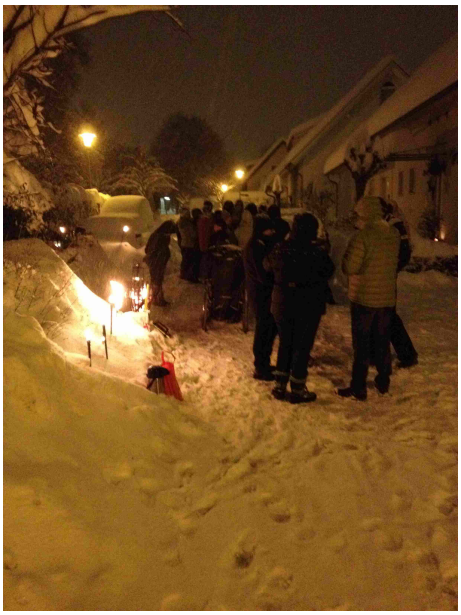
Willkommen zum „IBM Informix Newsletter“

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

wir hoffen, dass Sie gut ins Neue Jahr gestartet sind.
Nachdem der Winter sich mit viel Schnee und Kälte für ganze 4 Tage in der letzten Woche des alten Jahres doch noch gezeigt hat, ist inzwischen bereits fast der Frühling zu spüren.
Über die Feiertage haben wir einige Anregungen für sinnvolle PlugIns zum OAT erhalten. Vielen Dank dafür ! Wir werden die Vorschläge analysieren und nach und nach in Updates der PlugIns einbauen. Falls Sie auch Ideen haben, was im OAT noch hilfreich sein könnte, dann schreiben Sie uns.



Wie immer haben wir für Sie eine Reihe an Tipps und Tricks zusammengestellt.
Viel Spaß mit den Tipps der aktuellen Ausgabe.

Ihr TechTeam

TechTipp: Informix und NoSQL (6): Erste Schritte mit dem REST API

[Alle Beispiele wurden mit Informix Version 12.10.xC4 (auf Linux x86 64-bit) erprobt.]

In den letzten Ausgaben des Newsletters haben wir gesehen, wie wir NoSQL-Daten mit traditionellen SQL-Daten in Abfragen verknüpfen können, und dies sowohl mittels SQL als auch mit der Mongo-Shell über den JSON Wire Listener. In dieser Ausgabe betrachten wir, wie wir statt mit der Mongo-Shell über das REST API auf Daten zugreifen können.

Der Name "REST API" rührt daher, dass die Schnittstelle einen *REST-ful* Service bietet, d.h. der zugrundeliegende Dienst den Anforderungen einer **Re**presentational **S**tate **T**ransfer Architektur entspricht. Siehe hierzu auch folgenden englischen Wikipedia Artikel[1)]. Dort wird REST als eine Abstraktion der Architektur des World Wide Web beschrieben, die verlangt, dass eine Reihe von Einschränkungen auf die Komponenten, Verbindungen und Datenelemente angewandt wird. Die Einschränkungen sind folgende:

- **Client-server**: Client und Server sind voneinander getrennt.
- **Stateless**: Der Server speichert keine Information über den Zustand des Clients. Jede Anfrage eines Client an den Server enthält alle benötigte Information, die der Server für die Beantwortung braucht.
- **Cacheable**: Um den Datenaustausch zwischen Server und Client zu verringern, muss jede Antwort des Servers (implizit) definieren, ob der Client sie in seinem Cache speichern kann (zur späteren Wiederverwendung).
- **Layered System**: Ein Client kann nicht wissen, ob er direkt zu einem Server oder zu einem zwischengeschalteten System verbunden ist.
- **Uniform Interface**: Die einheitliche Schnittstelle, welche die Trennung von Client und Server ermöglicht.
- **Code on demand** (optional): Zur temporären Funktionserweiterung oder -anpassung kann ein Server dem Client ausführbaren Code schicken (z.B. JavaScript).

Das REST API für den Informix Server entspricht nicht nur den Anforderungen der REST Abstraktion, sondern auch ganz praktisch einem Web-Service, denn die einheitliche Schnittstelle verwendet das HTTP-Protokoll. Somit kann ein Internet-Browser tatsächlich als Client fungieren. Doch bevor wir das ausprobieren, müssen wir zuerst den nötigen Dienst auf der Seite des Informix Servers starten.

Wie für die MongoDB API, so erfolgt der Zugang über die REST API auch mit einem JSON Wire Listener. Das Binärprogramm ist dasselbe, ${INFORMIXDIR}/bin/jsonListener.jar, jedoch sind die Voraussetzungen und die Konfiguration für das REST API etwas anders:

- Voraussetzungen:
  - Java Runtime Environment (JRE):
    Für das REST API benötigt der JSON Wire Listener ein JRE der Version 1.7 oder neuer. Diese ist in der Version 12.10.xC4 des Informix Server leider noch nicht enthalten. Die vorhandene Version 1.6 genügt den Anforderungen des REST API nicht. Mehr Information zum IBM JRE 1.7 und dessen Download findet sich z.B. auf folgender developerWorks Seite: http://www.ibm.com/developerworks/java/jdk/index.html. (Ab Version 12.10.xC5 des Informix Server soll das IBM JRE der Version 1.7 mit dem Server ausgeliefert werden.)

  - Tomcat:
    Der JSON Wire Listener benötigt für das REST API Tomcat Version 8 (oder eine neuere Version davon). Tomcat Version 8 ist in der Datei ${INFORMIXDIR}/bin/nosql_sdk.zip enthalten, die mit dem Informix Server installiert wird.

- Konfiguration:

  Der einzige Unterschied zum JSON Wire Listener für das MongoDB API besteht im Parameter "type": statt "mongo" wird hier "rest" eingetragen.

Falls noch nicht auf dem System vorhanden muss also zuerst eine JRE der Version 1.7 (oder neuer) installiert werden. Je nach Platform und JRE sind hierfür die jeweiligen Anweisungen zu befolgen.

Für Tomcat packen wir die Datei ${INFORMIXDIR}/bin/nosql_sdk.zip so aus, dass der Inhalt im Verzeichnis ${INFORMIXDIR}/nosql_sdk landet.

Zur Konfiguration legen wir eine Datei namens
${INFORMIXDIR}/etc/restListener.properties
mit folgenden Inhalt an. Da wir sowohl die MongoDB API als auch die REST API unabhängig voneinander und gleichzeitig benutzen wollen, verwenden wir eine neue Portnummer, 26352 :

```
  listener.type=rest
  listener.port=26352
  security.sql.passthrough=true
  url=jdbc:informix-
sqli://localhost:24731/sysmaster:INFORMIXSERVER=mfu1;USER=informix
;PASSWORD=informix
```

Wie zuvor für den JSON Wire Listener des MongoDB API legen wir ein Start- und ein Stop-Script an. In diesen Scripts geben wir im Classpath zusätzliche Pfade an für die Nutzung von Tomcat. Die kompletten Dateien sehen dann folgendermassen aus:

- Start-Script rest_listener_start.sh:

```
#!/bin/sh

${INFORMIXDIR}/extend/krakatoa/jre/bin/java \
-cp ${INFORMIXDIR}/bin/jsonListener.jar:$
{INFORMIXDIR}/nosql_sdk/tomcat-embed-core.jar \
 com.ibm.nosql.informix.server.ListenerCLI \
-config ${INFORMIXDIR}/etc/restListener.properties \
-logfile /work/martinfu/731/restListener.log \
-start &

exit
```

- Stop-Script rest_listener_stop.sh:

```
#!/bin/sh

${INFORMIXDIR}/extend/krakatoa/jre/bin/java \
-cp ${INFORMIXDIR}/bin/jsonListener.jar:$
{INFORMIXDIR}/nosql_sdk/tomcat-embed-core.jar \
 com.ibm.nosql.informix.server.ListenerCLI \
-config ${INFORMIXDIR}/etc/restListener.properties \
-stop

exit
```

Schliesslich starten wir den JSON Wire Listener für das REST API, indem wir als Benutzer „informix“ das Start-Script ausführen.

Damit ist der Dienst bereit, und wir können einen Zugriff auf die Datenbank über das REST API ausprobieren. Weil das Protokoll HTTP ist, könnten wir für unsere erste Abfrage einen Internet-Browser, z.B. Firefox, benutzen und die Abfrage als URL (richtiger: als URI - Unified Resource Identifier) formulieren und an den JSON Wire Listener schicken. Um besser sehen und verstehen zu können, was da vor sich geht, benutzen wir jedoch statt einem Browser das Linux/UNIX Hilfsprogramm "cURL". Damit senden wir eine URI an den adressierten Dienst, das Hilfsprogramm empfängt die Antwort und stellt sie als primitiven Text dar - alles auf Kommandozeilenebene.

Mit dem Wissen, dass in unserer Informix Server Instanz die Datenbank "stores_demo" existiert, setzen wir unseren URI zusammen: IP-Addresse und Portnummer, die wir dem JSON Wire Listener als 'Horch-Addresse' definiert haben, und hängen mit einem '/' dann einfach den Datenbanknamen an, so dass wir als URI „localhost:26352/stores_demo“ erhalten. Diesen URI geben wir cURL als Kommandzeilenparameter:

```
% curl localhost:26352/stores_demo
["rating"]%
```

Als Antwort erhalten wir ein JSON-Array mit dem einzigen Element "rating" - die Collection in der Datenbank. Dies entspricht der Antwort, die wir auf das Kommando "show collection" von der Mongo Shell erhalten. Um besser zu verstehen, was vor sich gegangen ist, geben wir cURL zusätzlich den Parameter "-i", so dass wir auch Informationen zum HTTP-Header erhalten, den der JSON Wire Listener mit der Antwort an den Client (also cURL) schickt:

```
% curl -i localhost:26352/stores_demo
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: cursorid
Set-Cookie: informixRestListener.sessionId=95WO...; HttpOnly
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Thu, 1 Jan 2015 11:56:40 GMT

["rating"]%
```

Der interessanteste Teil des HTTP-Header der Antwort ist vielleicht der "Content-Type", denn wir sehen, dass der Inhalt der Antwort (nicht der HTTP-Header) weder vom Typ Text noch HTML ist, sondern JSON. Dies erklärt auch, warum die Ausgabe nach dem Array keinen Zeilenumbruch enthält. Wer noch etwas tiefer in die Ebene des HTTP-Protokolls hinabsteigen will, kann cURL statt "-i" die Option "-v" mitgeben. Damit ist auch zu sehen, was cURL als Anfrage an den Dienst schickt. In der Ausgabe sind Zusatzinformationen von cURL mit einem '*' gekennzeichnet, der HTTP-Header von cURL an den Server mit '>' und der HTTP-Header in der Antwort vom JSON Wire Listener mit '<' :

```
% curl -v localhost:26352/stores_demo
* About to connect() to localhost port 26352
*   Trying 127.0.0.1... connected
* Connected to localhost (127.0.0.1) port 26352
> GET /stores_demo HTTP/1.1
> User-Agent: curl/7.15.5 (x86_64-redhat-linux-gnu)
libcurl/7.15.5
```

```
>             OpenSSL/0.9.8b zlib/1.2.3 libidn/0.6.5
> Host: localhost:26352
> Accept: */*
>
< HTTP/1.1 200 OK
< Server: Apache-Coyote/1.1
< Access-Control-Allow-Credentials: true
< Access-Control-Expose-Headers: cursorid
< Set-Cookie: informixRestListener.sessionId=vSUW...; HttpOnly
< Content-Type: application/json;charset=UTF-8
< Transfer-Encoding: chunked
< Date: Wed, 1 Jan 2014 12:01:15 GMT
* Connection #0 to host localhost left intact
* Closing connection #0
["rating"]%
```

cURL sendet einen GET-Request für die Resource "stores_demo", der JSON Wire Listener antwortet mit "200 OK" und sendet den Inhalt als JSON-Daten.

Versuchen wir also, in derselben Weise eine Anfrage zur Collection "rating" selbst zu stellen, indem wir einfach mit einem zusätzlichen '/' "rating" an den URI anhängen, und damit wieder cURL aufrufen:

```
% curl localhost:26352/stores_demo/rating
[{"_id":{"$oid":"5487356fcb202e2fbae7d2ce"},"catalog_num":1001
7.0,"rating_value":7.0},{"_id":{"$oid":"5487356fcb202e2fbae7d2
cf"},"catalog_num":10017.0,"rating_value":9.0},{"_id":{"$oid":
"5487356fcb202e2fbae7d2d0"},"catalog_num":10019.0,"rating_valu
e":7.0},{"_id":{"$oid":"5487356fcb202e2fbae7d2d1"},"catalog_nu
m":10017.0,"rating_value":5.0},{"_id":{"$oid":"5487356fcb202e2
fbae7d2d2"},"catalog_num":10017.0,"rating_value":8.0},{"_id":{
"$oid":"5487356fcb202e2fbae7d2d3"},"catalog_num":10019.0,"rati
ng_value":3.0},{"_id":{"$oid":"5487356fcb202e2fbae7d2d4"},"cat
alog_num":10017.0,"rating_value":7.0},{"_id":{"$oid":"5487356f
cb202e2fbae7d2d5"},"catalog_num":10017.0,"rating_value":8.0},{
"_id":{"$oid":"5487356fcb202e2fbae7d2d6"},"catalog_num":10017.
0,"rating_value":4.0},{"_id":{"$oid":"5487356fcb202e2fbae7d2d7
"},"catalog_num":10019.0,"rating_value":8.0},{"_id":{"$oid":"5
487356fcb202e2fbae7d2d8"},"catalog_num":10019.0,"rating_value"
:3.0},{"_id":{"$oid":"5487356fcb202e2fbae7d2d9"},"catalog_num"
:10017.0,"rating_value":8.0}]%
```

Als Antwort erhalten wir wieder JSON-Daten, die ein Array mit den Dokumenten der Collection "rating" enthalten. Das Ganze ist zwar nicht besonders leserlich, da keinerlei Formatierung stattfindet, sondern alles ohne Zeilenumbruch und Leerzeichen geschickt wird. Bei näherem Hinsehen können wir jedoch die Struktur der Dokumente und die Datenwerte erkennen. Wir haben also durch Absenden einer einfachen URI die Daten der gewünschten Collection in der Datenbank "stores_demo" erhalten. Und all dies mit einem Standard-Client, der nur das gängige HTTP-Protokoll benutzt.

Da JSON für JavaScript Object Notation steht, liegt es nun nahe, die vom JSON Wire Listener erhaltenen JSON-Daten mit einem JavaScript zu verarbeiten, was mit einem Internet-Browser, der JavaScript ausführen kann und darf, nicht allzu schwierig sein sollte. Den Schritt zu einem ersten, kleinen Java-Programm werden wir in einer der nächsten Ausgaben des Newsletters angehen.

1) Wikipedia Artikel zu REST: http://en.wikipedia.org/wiki/Representational_state_transfer

TechTipp: Owner von Datenbankobjekten – Size does matter !

Namen von Objekten wie z.B. Tabellen, Indices, Synonyme, … sind bei INFORMIX per Default nicht case-sensitiv, falls nicht DELIMIDENT(1) als Umgebungsvariable gesetzt wurde. Wie sieht es aber mit dem Besitzer des Objektes aus ?
Informix erlaubt die Abfrage von Tabellen aller „Owner“, auch wenn nur der Tabellenname angegeben wird. Ausnahme hierbei ist der Mode ANSI, der nur die Tabellen zum eigenen Benutzer (bei anderen Datenbanken auch Schema genannt) findet, so lange nicht explizit der Owner mit angegeben wird, da im Mode ANSI der selbe Tabellenname mit unterschiedlichen Ownern genutzt werden kann.
Abfragen, die über ODBC oder JDBC kommen, stellen oft das Schema dem Tabellennamen voraus, damit die identische Syntax auf vielen unterstützten Plattformen lauffähig ist. Wird ein falscher „Owner“ bei der Abfrage mitgegeben, so erhält man die Fehlermeldung „-206 Table not found“.
Der Konkrete Anlass für diesen TechTipp war nun der Umstand, dass auch die Gross- und Kleinschreibung beim Owner eine Rolle spielt und die Eingabe teilweise implizit durch die Datenbank verändert wird.

So kann z.B. eine Tabelle, die mit dem Statement
„create table xxx (f1 int)“ angelegt wurde, erfolgreich über folgende Statements abgefragt werden:

```
select * from xxx;
select * from XXX;
```

Anders verhält sich dies beim Owner der Tabelle.
Die selbe Abfrage nun mit explizit angegebenem Owner:
(Logname „anna“ hat die Tabelle erstellt)

```
select * from anna.testtab;
select * from ANNA.testtab;
select * from "anna".testtab;
select * from "ANNA".testtab;
#                     ^
#  206: The specified table (ANNA.testtab) is not in the database.
```

Wird der Owner beim Anlegen der Tabelle nicht explizit angegeben, so wird der „logname“ in Kleinbuchstaben als „Owner“ hinterlegt.
Wird hingegen ein „Owner“ explizit angegeben, so hängt die Speicherung des „Owners“ davon ab, ob die Angabe in Anführungszeichen erfolgt, und ob die Datenbank im Log Mode ANSI erstellt wurde.
Am besten lässt sich dies am folgenden Beispiel zeigen, das Sie selbst so auf jeder Informix Instanz testen können:

Wir legen eine Datenbank an und erstellen Tabellen mit unterschiedlicher Angabe des Owners:

Der Test startet immer von „neutralem Grund“, also in der Datenbank „sysmaster“.

```
database sysmaster;
drop database if exists xxx;
create database xxx with log;
     --für den Testfall „with log“

-- create database xxx with log mode ansi;
     --für den Testfall „mode ANSI“

create table "anna".tab1_quote_klein (f1 int);
create table "ANNA".tab2_quote_gross (f1 int);
create table "Anna".tab3_quote_initcap (f1 int);
create table tab4_ohne (f1 int);
create table anna.tab5_klein (f1 int);
create table ANNA.tab6_gross (f1 int);
create table Anna.tab7_initcap (f1 int);
```

und fragen anschliessend den Owner und die Tabelle aus „systables“ ab:

```
select tabname[1,18], owner
from informix.systables
where tabid > 99 order by 1;
```

Database with mit und ohne logging:

```
tabname             owner
tab1_quote_klein    anna
tab2_quote_gross    ANNA
tab3_quote_initcap  Anna
tab4_ohne           anna
tab5_klein          anna
tab6_gross          anna
tab7_initcap        anna
```

Wir sehen, dass bei der Angabe des „Owners“ ohne Anführungszeichen immer der „logname“ in **Kleinbuchstaben** gespeichert wird.

Database with log mode ANSI:

```
tabname             owner
tab1_quote_klein    anna
tab2_quote_gross    ANNA
tab3_quote_initcap  Anna
tab4_ohne           anna
tab5_klein          ANNA
tab6_gross          ANNA
tab7_initcap        ANNA
```

Beim Log Mode ANSI wird der angegebene Name immer in **Grossbuchstaben** gespeichert, wenn dieser nicht in Anführungszeichen angegeben wird. Ohne explizite Angabe wird weiterhin der „logname“ in Kleinbuchstaben genutzt.
Es macht somit einen grossen Unterschied, ob die Datenbank im Mode ANSI erstellt ist oder nicht, und ob der angegebene Owner beim Erstellen der Tabelle in Anführungszeichen gesetzt wurde.
Für den Fall der Datenbank im Mode ANSI, besteht zudem die Option, die Umgebungsvariable ANSIOWNER zu setzen. Dieser Fall wird im folgenden TechTipp behandelt.

(1) DELIMIDENT wurde im Informix Newsletter November 2006 behandelt.

TechTipp: Environments - ANSIOWNER

Im vorhergehenden TechTipp wurde gezeigt, wie der angegebene Owner beim Erstellen von Objekten gespeichert wird. Im Fall einer Datenbank im Mode ANSI kann durch die gesetzte Umgebungsvariable ANSIOWNER die Eingabe des Owners ohne Anführungszeichen auf das Verhalten gesetzt werden, als wären die Anführungszeichen verwendet worden.
Der Test aus dem vorhergehenden Artikel zeigt dann folgende Ausgabe:

Database with log mode ANSI + ANSIOWNER=1:

```
tabname             owner
tab1_quote_klein    anna
tab2_quote_gross    ANNA
tab3_quote_initcap  Anna
tab4_ohne           anna
tab5_klein          anna
tab6_gross          ANNA
tab7_initcap        Anna
```

TechTipp: Datablades mit Migrationshintergrund

Wird eine Datenbank migriert, in der Datablades wie z.B. TimeSeries, Spatialdaten, BasicTextSearch, … verwendet wurden, dann empfiehlt es sich, nach der Migration die Instanz noch einmal neu zu starten, nachdem implizite Anpassungen durchgeführt wurden, die von den Tasks mit dem Namen „autoreg migrate“ erledigt werden.
Hierbei werden z.B. die Pfadnamen zu den externen Prozeduren und Funktionen in der Tabelle „sysprocedures“ aktualisiert. Damit diese Änderungen nach der Aktualisierung auch im Procedure-Cache korrekt enthalten sind, empfiehlt sich ein Neustart der Instanz, sobald im OAT zu sehen ist, dass der „autoreg migrate“ auf allen Datenbanken ausgeführt wurde:

autoreg migrate ibm	786	19.52	15343.28	2015-01-15 08:23:54	✓
autoreg migrate kalu	786	18.81	14792.33	2015-01-15 08:23:52	✓
autoreg migrate stores	346	20.59	7127.30	2015-01-15 08:23:53	✓
autoreg migrate xxx	6	1.13	6.78	2015-01-15 08:23:36	✓

Zudem sollte man ggf. die Metadaten vom bisherigen in das Neue INFORMIXDIR kopieren, wie dies z.B. bei Spatialdaten unbedingt erforderlich ist.

TechTipp: Informix für Himbeeren

Nachdem Informix schon seit einiger Zeit für die Apple-Plattform verfügbar ist, gibt es mit der Developer-Edition 12.10.UC4DE den Informix-Server nun auch für Raspberry Pi. Allerdings sollte man nicht Äpfel mit Himbeeren vergleichen, denn Raspberry Pi ist ein Einplatinenrechner in Kreditkartengrösse zu einem Stückpreis gut unter 50 EUR (aktuell im Web gesehen für 30€).

Das Betriebssystem für Informix ist Raspbian GNU/Linux 7 (armhf) und läuft mit dem ARMv6 Befehlssatz auf der Familie der ARM11-Prozessoren des Raspberry Pi. Gebaut und getestet wird diese Informix-Version auf einem Raspberry Pi Modell B+. Mit dem minimalen Footprint von 64 MB bleibt von den vorhandenen 512 MB Speicher noch genug für einen brauchbaren Bufferpool übrig. Als "Festpaltte" dient eine SDHC-Speicherkarte.

Mit zusätzlicher Hardware wie z.B. dem Gertboard kann auch Peripherie angesteuert werden, z.B. Sensoren, Schalter und Elektromotoren. Damit kann Raspberry Pi als Baustein für das IoT (Internet der Dinge) dienen. Der Informix Server ist so in nächster Nähe zu den Datenquellen, z.B. um Zeitreihendaten quasi "vor Ort" zu sammeln, zu speichern und weiter zu verarbeiten.

Download Link für die Informix Developer Edition:
https://www.software.ibm.com/webapp/iwm/web/reg/pick.do?source=ifxids&S_TACT=109HF16W&lang=en_US

TechTipp: Vanilla beim Check – Kein Grund zur Sorge

Der Konsistenzcheck einer Datenbank mittels „oncheck -cDI <dbname>“ prüft nicht die Integrität von funktionalen Indexen. Stattdessen wird eine Warnung ausgegeben, dass es sich um einen „Non vanilla index“ handelt, der nicht überprüft werden kann.
Beispiel: Auf der Tabelle „person“ wurde ein funktionaler Index „i_person_toup“ auf das Feld „name“ als Funktion „to_upper(name)“ erstellt, der die caseinsensitive Suche beschleunigen soll. Der Check der Tabelle bringt dann folgende Warnung:

WARNING: index check requires a s-lock on tables whose lock level is page.
Index i_person_toup: **Non vanilla index. data crosschecking not enabled**

WebTipp: Informix Blog von Fernando Nunes

In Web finden sich einige interessante Blogs zu Informix. Diesmal wollen wir auf den Blog von Fernando Nunes hinweisen, der seine Artikel in englischer und portugiesischer Sprache veröffentlicht und durch seinen Hintergrund auch einen Fokus auf den sehr interessanten Markt Südamerika hat. Sie finden diesen Blog unter:
http://informix-technology.blogspot.de/

WebTipp: A SQL database for sensor and JSON data on Raspian

Ein Artikel zu INFORIMIX und Raspberry Pi ist auch auf den Seiten der Raspberry Community zu finden:
http://www.raspberrypi.org/forums/viewtopic.php?f=37&t=97199

Anmeldung / Abmeldung / Anmerkung

Der Newsletter wird ausschließlich an angemeldete Adressen verschickt. Die Anmeldung erfolgt, indem Sie eine Email mit dem Betreff „**ANMELDUNG**“ an **ifmxnews@de.ibm.com** senden.
Im Falle einer Abmeldung senden Sie „ABMELDUNG“ an diese Adresse.

Das Archiv der bisherigen Ausgaben finden Sie zum Beispiel unter:

- http://www.iiug.org/intl/deu
- http://www.iug.de/index.php?option=com_content&task=view&id=95&Itemid=149
- http://www.informix-zone.com/informix-german-newsletter
- http://www.drap.de/link/informix
- http://www.nsi.de/informix/newsletter
- http://www.cursor-distribution.de/index.php/aktuelles/informix-newsletter
- http://www.listec.de/Newsletter/IBM-Informix-Newsletter/View-category.html
- http://www.bereos.eu/software/informix/newsletter/

Die hier veröffentlichten Tipps&Tricks erheben keinen Anspruch auf Vollständigkeit. Da uns weder Tippfehler noch Irrtümer fremd sind, bitten wir hier um Nachsicht falls sich bei der Recherche einmal etwas eingeschlichen hat, was nicht wie beschrieben funktioniert.

Die Autoren dieser Ausgabe

Gerd Kaluzinski IT-Specialist Informix Dynamic Server und DB2 UDB
IBM Software Group, Information Management
gerd.kaluzinski@de.ibm.com +49-175-228-1983

Martin Fuerderer IBM Informix Entwicklung, München
IBM Software Group, Information Management
martinfu@de.ibm.com

Markus Holzbauer IBM Informix Advanced Support
IBM Software Group, Information Management Support
holzbauer@de.ibm.com

Die Versionsinfo stammt aus dem Versions-Newsletter der CURSOR Software AG
http://www.cursor-distribution.de/download/informix-vinfo

Sowie unterstützende Teams im Hintergrund.

Fotonachweis: Gerd Kaluzinski (Schneebar in Lindau)