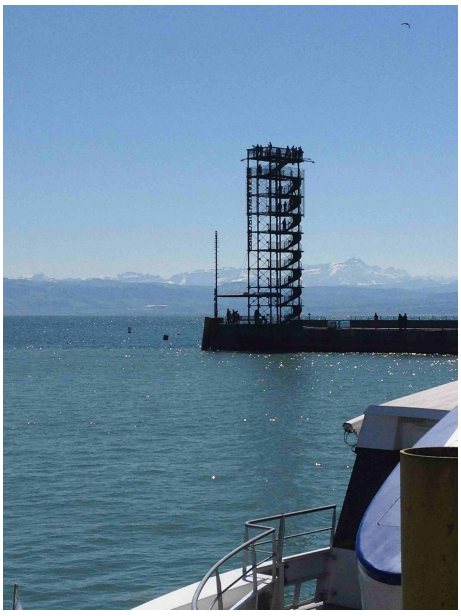
Willkommen zum „IBM Informix Newsletter“

Inhaltsverzeichnis

Aktuelles

Liebe Leserinnen und Leser,

wir hoffen, dass Sie das tolle Wetter im Frühling nutzen konnten um Kraft zu tanken für die vielen Veranstaltungen, die im Mai auf Sie warten. In diesem Newsletter finden Sie Details zu einigen Neuerungen der Version 12.10.FC5, sowie ein nachvollziehbares Beispiel zu NoSQL mit INFORMIX.
Ein neues Feature wurde in der Vergangenheit oft von Administratoren nachgefragt: „Wie kann ich eine Session davon abhalten meinen Betrieb massiv zu stören ?“ Die neuen Session Limits sind hier sicher ein guter Weg um Sessions, die dem Gesamtsystem schaden könnten, in Schranken zu verweisen.



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

Ihr TechTeam

TechTipp: Übersicht der Änderungen in 12.10.FC5

Mit dem Release 12.10.FC5 kamen einige Neuerungen in den INFORMIX Server. Da INFORMIX neben vielen treuen Bestandskunden auch zahlreiche Neukunden gewinnen konnte, die sich oft auf Grund der umfangreichen Möglichkeiten bei TimeSeries und JSON für INFORMIX entschieden haben, ist es nicht verwunderlich dass ein grosser Teil der neuen Features in diesem Umfeld zu finden ist.

Anbei eine Übersicht über die Neuerungen:

Migration:

Rolling Upgrade in einer HDR/RSS Umgebung (z.B. 12.10.FC4 auf FC5)
bei dem nach und nach die Instanzen migriert werden.

Installation:

Support für Java 7
Verbessertes Logging und Debugging bei der Installation, das mit der Option -DDEBUG_LEVEL zwischen 1 und 9 gewählt werden kann.
Mit -DOVERWRITE_PRODUCT=TRUE kann bei einer Silent-Installation eine bisherige Installation ersetzt werden, ohne dass diese zuvor gelöscht werden muss.

Administration:

Limitierung von Sessions durch neuen Parameter in der ONCONFIG
(Details weiter unten in einem eigenen Artikel)

Kontrolle der Resourcen für Tenants im Bezug auf TempSpace und Memory.
Einschränkung der OAT-Ansichten auf Tenants.

Backup:

Das Limit für Tapesize wurde auf 9ZB erhöht (9223372036854775807 KB)

JDBC:

Neue Sprachumgebung für Estland und Litauen

SQL:

Die Umgebungsvariablen AUTO_REPREPARE und IFX_AUTO_REPREPARE wurden um zusätzliche Optionen erweitert.

Korrelierte Ausdrücke bei Aggregate sind nun in Subqueries möglich.

JSON compatibility:

Hier gab es eine ganze Reihe an Erweiterungen, die wir in einer der kommenden Ausgaben des INFORMIX Newsletters ausführlich vorstellen werden.

TimeSeries Loader:

Das Laden von JSON Dokumenten ist nun unterstützt, zudem wurden einige Verbesserungen im Loader eingebaut die Logging und „autocreate“ betreffen.

TimeSeries Administration:

Neue Funktionen zur Administration von TimeSeries stehen zur Verfügung. Diese werden wir in einer der kommenden Ausgaben vorstellen.

TimeSeries Abfrage:

Patternsuche in TimeSeries wurde ermöglicht.
Die Clip Funktion wurde erweitert.

Spatial und TimeSeriesdaten verbinden:

Bewegte Objekte können in einer „Spatialtemporal-Suche“ angefragt werden. Dies stellt eine Verbindung der TimeSeries zu den Spatialdaten dar.
Da dieses Thema spannend ist, planen wir hierzu einen eigenen Beitrag in einer den kommenden Ausgaben.

Bei der Datenbank sysmasters wurden keine Änderungen vorgenommen gegenüber der Version 12.10.FC4.
Die Datenbanken sysutils und sysuser sind ebenfalls unverändert.
In der Datenbank sysadmin wurden die Werte im Task „mon_table_profile“ von Integer auf BigINT geändert, damit auch bei Datenbanken mit hohem Durchsatz die Werte korrekt dokumentiert werden können.

Die Vielzahl an Änderungen zum INFORMIX **Warehouse Accelerator** werden im folgenden Artikel beschrieben.

TechTipp: Informix Warehouse Accelerator: Neu in 12.10.FC5

Seit dem 26. März 2015 ist die neue Version 12.10.xC5 des Informix Servers elektronisch erhältlich (sog. 'eGA'). Die kostenlose Developer oder Time Limited Edition gibt es unter folgender Adresse:

https://www14.software.ibm.com/webapp/iwm/web/preLogin.do?source=ifxids

Diese neue Version enthält auch einige Erweiterungen des Informix Warehouse Accelerators (IWA). Folgend jeweils eine kurze Beschreibung:

- **UNITS Operator für MONTH und YEAR**

  IWA unterstützt nun den UNITS Operator für MONTH und YEAR. Ein üblicher Anwendungsfall des UNITS Operators ist bei Datumsarithmetik, z.B.:

  ```
  SELECT lead_time + 5 UNITS DAY FROM manufact;
  ```

- **Voreinstellung UNITS DAY**

  Eine Informix-Spezialität ist, dass der Operator UNITS DAY nicht angegeben werden muss. Dies wird nun auch von IWA unterstützt. In diesem Sinne ist die folgende Abfrage gleichbedeutend mit dem obigen Beispiel:

  SELECT lead_time + 5 FROM manufact;

- **Unterstützung der Funktion SQRT**

  IWA unterstützt nun die mathematische Funktion SQRT(), also die Quadratwurzel. Die Funktion gibt die positive Quadratwurzel des Arguments zurück, wobei das Argument ein positiver numerischer Ausdruck sein muss. Beispiel:

  ```
  SELECT SQRT(9) FROM angles;
  ```

- **Operatoren CURRENT und SYSDATE**

  Die Operatoren CURRENT und SYSDATE werden nun von IWA unterstützt. Der Operator CURRENT liefert einen Wert vom Typ DATETIME YEAR TO FRACTION(3) des derzeitigen Datums mit Uhrzeit. Der Operator SYSDATE ist identisch mit dem Operator CURRENT, allerdings mit dem Unterschied, dass ein Wert von SYSDATE vom Typ DATETIME YEAR TO FRACTION(5) ist. Beispiel:

  ```
  SELECT * FROM tab1 WHERE datetime_column < CURRENT;
  ```

Neues Kommando "ondwa listmarts"

Für das Hilfsprogramm "ondwa" gibt es ein neues Kommando "listmarts". Dies ist hilfreich für den Administrator, da es in der Umgebung des IWA ausgeführt wird und existierende Data Marts der verschiedenen Acceleratoren anzeigt. Beispiel:

```
$ ondwa listmarts
Found 4 marts:
mart="cq_sqrt", martid="151", accelerator="IDS.dwa1",
state="ACTIVE", epoch="1", lastload="2014-12-08
11:23:16.788831"
mart="iwadb_mart1", martid="152", accelerator="IDS.dwa1",
state="ACTIVE", epoch="3", lastload="2015-01-09
17:44:40.822332", lastupdate="2015-01-13 16:43:49.048554"
mart="iwadb_mart2", martid="153", accelerator="IDS.dwa1",
state="LOAD PENDING", epoch="0"
mart="iwadb_mart2", martid="154", accelerator="IDS.dwa2",
state="LOAD PENDING", epoch="0"
```

TechTipp: Environments: IFX_NOZEROMDY

Ein Datum mit dem Wert „00/00/0000“ bzw. „00.00.0000“ bei DBDATE=DMY4. wird üblicherweise vom Datenbankserver als NULL interpretiert und auch so abgespeichert. Soll statt dem Nullvalue eine Fehlermeldung ausgegeben werden, so kann dies mit der Umgebungsvariablen IFX_NOZEROMDY=1 erzwungen werden. Die Umgebungsvariable muss dabei vor dem Start der Instanz gesetzt sein und gilt für alle Sessions der Datenbank.
Auf Werte vom Typ DATETIME hat diese Umgebungsvariable keine Auswirkung.

TechTipp: ONCONFIG – SESSION_LIMIT_xxx

Bisher war es nicht möglich einzelne Sessions davon abzuhalten, Unmengen an Sperren zu nutzen, lange andauernde Transaktionen zu starten (so lange diese nicht LTXHWM überschritten haben), den TempDBSpace zu füllen oder aussergewöhnlich viel Memory zu verwenden.
Mit Version 12.10.FC5 kann der Administrator über Parameter in der ONCONFIG allgemeinverbindliche Limits für Sessions definieren. Ausgenommen davon sind User mit Administrationsrechten.

Die neuen Parameter sind:

- SESSION_LIMIT_LOCKS, mit dem man die Anzahl der Sperren auf einen Wert zwischen 500 und 2 Milliarden (Maxint) begrenzen kann. Default: 0 (Off)
- SESSION_LIMIT_MEMORY, mit dem der Speicher, den eine Session belegen kann, auf einen Wert zwischen 5MB und 2048GB begrenzt werden. Die Eingabe erfolgt in KB, der Default ist 0 (Off).
- SESSION_LIMIT_TEMPSPACE, mit dem man die Verwendung von Platz in den TempDBSpaces auf einen Wert zwischen 20MB und 2048GB einschränken kann. Die Eingabe erfolgt in KB, der Default ist 0 (Off).
- SESSION_LIMIT_LOGSPACE begrenzt in den Logischn Logs der den maximalen Platz, der innerhalb einer Transaktion belegt werden darf. Der Wert kann zwischen 5MB und 2048GB eingestellt werden und wird in KB angegeben. Der Default ist 0 (Off).
- SESSION_LIMIT_TXN_TIME gibt die maximale Zeit in Sekunden an, die eine Transaktion dauern darf. Mögliche Werte sind zwischen 60 und 2 Milliarden (Maxint). Die Eingabe erfolgt in Sekunden. Der Default ist 0 (Off).

Trifft eine Session auf eines der Limits, so wird eine Fehlermeldung ausgegeben, die den Grund beschreibt (ohne allerdings auf das Session Limit hinzuweisen).
Beispiel:

```
211: Cannot read system catalog (sysprocedures).
 134: ISAM error: no more locks
```

Limits einer Sessions sind im „onstat -g ses <session_id>“ am Ende der Ausgabe zu sehen, falls Limits vorhanden sind:

```
Session Limits
                Limit       Current
Locks           500         213
Memory(KB)      5120        512
Log Space(KB)   10000       0
Temp Space(KB)  204800      16
Txn Time(s)     60          42
```

Alle Limits können mittels „onmode -wf“ im laufenden Betrieb gesetzt bzw. geändert werden. Im Messagelog werden die Änderungen protokolliert:

```
12:05:38  Value of SESSION_LIMIT_LOCKS has been changed to 500.
12:06:02  Value of SESSION_LIMIT_MEMORY has been changed to 5120
kilobytes.
12:06:20  Value of SESSION_LIMIT_TEMPSPACE has been changed to
204800 kilobytes.
12:06:57  Value of SESSION_LIMIT_LOGSPACE has been changed to
10000 kilobytes.
12:07:10  Value of SESSION_LIMIT_TXN_TIME has been changed to 60
kilobytes.
```

(Anmerkung der Redaktion: Wir messen Zeit weiterhin in Sekunden, Minuten, Stunden, … statt in Kilobytes und haben dieses kosmetische Problem gemeldet).

Trifft eine Session auf ein Limit, so ist dies ebenfalls im Messagelog zu sehen:

```
04/14/15 12:13:42  Session SID=151 User UID=1000 NAME=kalu
PID=31309 has exceeded the session limit of 500 locks.
```

Ein weiteres Beispiel zur Transaktionszeit:

```
begin work;
update customer set lname = upper(lname) where customer_num =
104 ;
 1 row(s) updated.
```

… nun vergeht einige Zeit ...

```
commit
  458: Long transaction aborted.
 12204: RSAM error: Long transaction detected.
```

Der „onstat -g ses“ listet am Ende der Ausgabe zudem die letzten 20 Überschreitungen der Limits auf:

```
Session Limits
                 Limit      Current
Txn Time(s)      60         0

Last 20 Sessions Terminated
Ses ID Username  Hostname     PID    Time              Reason
206    kalu      kalu3        27539  04/14/2015.13:55
                                     session limit txn time (85 s)
206    kalu      kalu3        27539  04/14/2015.13:59
                                     session limit txn time (116 s)
Total Terminated 2
```

Die Zuordnung der Resourcen auf Tenants hat Vorrang gegenüber den Limitierungen der Sessions mittels Parametern in der ONCONFIG.

TechTipp: Zugriffsrechte in der Datenbank sysadmin verwalten

Die Datenbank sysadmin ist nach der Installation so eingerichtet, dass nur der User „informix“ das Recht „connect“ besitzt und sich zu dieser Datenbank verbinden darf. Um weiteren Usern Rechte zu vergeben, kann der Task „grant admin“ genutzt werden. Der Aufruf lautet:

execute function task('grant admin','<user>','<privilege group>');

Beispiel: Dem User „kalu“ sollen die Aufrufe zu Sicherung und Restore mittels Taskfunktionen zum „onbar“ erlaubt werden:

```
execute function task('grant admin','kalu','bar');
(expression)  User kalu permissions have been set to bar.
```

Folgende Rechte können einem User mittels „grand admin“ zugeteilt werden:

- **ADMIN**: Alle SQL Admin Funktionen.
- **BAR:** Backup und Restore.
- **FILE**: Manage Messagelog und Anzeite von Dateiinformationen.
- **GRANT**: Grant und Revoke von Berechtigungen.
- **HA**: Funktionen zur Hochverfügbarkeit.
- **MISC**: Administration des Datenbankservers.
- **MONITOR**: Aufruf aller SQL API Funktionen, die nur anzeigen und nicht ändern.
- **OPERATOR**: Alle SQL API Funktionen ausser GRANT und REVOKE.
- **REPLICATION**: Funktionen zur Enterprice Replikation (CDR).
- **SQL**: Funktionen die sich auf SQL Statements beziehen.
- **SQLTRACE**: Funktionen zum SQL Trace.
- **STORAGE**: Verwaltung von Storagepools, DBSpaces und Chunks.
- **TENANT**: Funktionen zur Verwaltung von Resourcen für Tenants.
- **WAREHOUSE**: Aufruf von Informix® Warehouse Accelerator administration tools.

TechTipp: Informix und NoSQL (7): "Hello, database!" - Ein kleines JavaScript

[Alle Beispiele wurden mit Informix Version 12.10.xC5 (auf Linux x86 64-bit) erprobt.]

Nach dem Aufsetzen des JSON Wire Listener für das REST API waren wir schon in der Lage, auf Daten in der Collection "rating" der Datenbank "stores_demo" zuzugreifen. Allerdings, wie wir in der Januar-Ausgabe des Newsletters feststellen mussten, sind die Daten, so wie sie vom REST API im JSON-Format geliefert werden, alles andere als leicht lesbar. Eine Weiterverarbeitung der Daten auf der Client-Seite ist angebracht. Was wäre besser geeignet als ein JavaScript-Programm, um Daten in der JavaScript Object Notation aufzubereiten?

Unser erster Versuch ist ein sehr minimalistisches JavaScript-Programm, eingebettet in HTML, so dass es ohne weiteres in einem Internetbrowser benutzt werden kann. Die grundlegenden Voraussetzungen dafür sind:

- Der Browser muss *dazu in der Lage sein*, JavaScript-Code auszuführen. Heutzutage können das die meisten Browser.
- Dem Browser muss *es erlaubt sein*, JavaScript-Code auszuführen. Ein Browser kann gegenteilig konfiguriert sein, meist aus Sicherheitsgründen. In diesem Fall muss die Konfiguration des Browsers geändert werden.
- Der Browser kann zur IP-Adresse und Portnummer des JSON Wire Listeners eine Netzwerkverbindung herstellen. Fehlende Netzwerkkonnektivität oder Firewalls könnten dies evtl. verhindern.

Der Einfachheit halber betten wir den JavaScript-Code in ein umrahmendes HTML-Dokument ein. Wir können dann den Browser das HTML-Dokument 'anzeigen' lassen. Dabei ist es unwichtig, wo sich das HTML-Dokument befindet. Es kann lokal sein, so dass der Browser es ohne einen Webserver finden kann. Oder es kann auf einer anderen Maschine sein, von wo der Browser das Dokument von einem Webserver anfordern kann. Im ersten Fall beginnt die URL für das lokale Dokument mit file:///..., wobei ... für den Pfad und Dateinamen des Dokuments steht. Im zweiten Fall bezeichnet die URL die Adresse des Webservers, normalerweise mit http:// beginnend.

Schauen wir uns nun Stück für Stück den Inhalt des HTML-Dokuments an. Zur Referenz finden Sie das komplette Dokument mit Zeilennummern am Ende dieses Artikels.

Das HTML Gerüst des Dokuments beginnt mit dem HTML Header, gefolgt vom <body>-tag, das den Anfang des Dokumenteninhalts markiert:

```
1   <html>
2   <head>
3   <title>rest_api_01</title>
4   </head>
5   <body>
```

Um sichtbare Elemente im HTML-Dokument möglichst auf ein Minimum zu beschränken, haben wir in der Hauptsache nur ein HTML Form-Element, das einen einzelnen HTML Input-Button enthält:

```
39   <form>
40     <input type="button" value="Go" onClick="getTableData();">
41   </form>
```

Im Browser wird dieser Input-Button mit seinem Wert, also "Go", bezeichnet. Mit dem Attribut "onClick" wird, wie der Name schon verrät, spezifiziert, was geschehen soll, wenn der Knopf betätigt wird. Wir geben hierfür den Namen einer Funktion "getTableData()" an. Diese Funktion werden wir im eingebetteten JavaScript-Code definieren.

Als nächstes fügen wir ein HTML Division-Element ein. Vorerst bleibt es leer, aber wir geben diesem Element eine explizite ID, "div_data01". Im JavaScript-Code können wir dann diese ID benutzen, um das Division-Element zu referenzieren:

```
43   <hr />
44   <div id="div_data01">
45
46   </div>
47   <hr />
```

Nur damit wir Anfang und Ende dieses Division-Elements besser sehen - besonders da es zu Beginn leer ist - fügen wir vor und nach dem Division-Element mit einem <hr>-tag jeweils eine horizontale Linie ein.

Mit </body> und </html> schliessen wir das HTML-Dokument ordnungsgemäss ab:

```
49   </body>
50   </html>
```

Wir haben nun ein komplettes und syntaktisch richtiges HTML-Dokument, das wir den Browser anzeigen lassen können. Der minimalistische Inhalt ist der Input-Button "Go" und zwei horizontale Linien, wie in Abbildung 1 dargestellt. Wir können den Knopf drücken, aber aus gutem Grund passiert nichts, denn wir haben die Funktion "getTableData()", die der Browser beim Klick aufrufen soll, noch nicht implementiert.

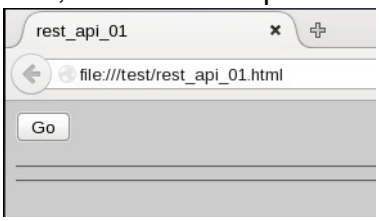


Abb. 1: Ausgangsansicht des HTML-Dokuments im Browserfenster

Um JavaScript-Code in ein HTML-Dokument einzubetten, müssen wir dem Browser mitteilen, dass dies ein Script ist, das JavaScript-Code enthält. Andernfalls würde der Browser den Code als HTML auffassen und versuchen, diesen darzustellen, was im Browserfenster sehr wahrscheinlich verstümmelt aussieht und sicherlich unseren Knopf nicht funktionsfähig macht. Daher umschliessen wir den JavaScript-Code folgendermassen mit <script>-tags:

```
  7   <script language="JavaScript">
      ...
 37   </script>
```

Sämtlichen JavaScript-Code platzieren wir zwischen diese zwei tags. Wir wollen die Funktion implementieren, die aufgerufen werden soll, wenn der Input-Button gedrückt wird. Diese Funktion muss den Namen "getTableData" haben und soll keine Parameter erwarten, denn so benutzen wir sie in der Definition des Input-Buttons. Da wir erwarten, dass beim Click auf den Input-Button etwas sichtbar im Browserfenster erscheint, sollte die Funktion eine Ausgabe produzieren. Dies bewerkstelligen wir, indem wir einen String ("divString") zusammensetzen, und diesen dann in Zeile 34 dem Division-Element des HTML-Dokuments als neuen Inhalt zuweisen. Dies ist die Stelle, wo wir die ID "div_data01" benutzen, um das Division-Element zu referenzieren:

```
 20     function getTableData() {
 21       var i = 0;
 22       var divString = '';
          ...
 34       document.getElementById('div_data01').innerHTML =
divString;
 35     }
```

Im übrigen soll die Funktion die Daten holen und diese in die Variable "divString" füllen. Um die Daten zu holen, müssen wir eine URI erstellen, ähnlich wie bei der Benutzung des Linux Hilfsprogramms "cURL" (siehe auch Januar-Ausgabe des Newsletters).

Wir benutzen dazu die Variable "baseUri", hängen mit einem vorangestellten '/' den Namen "rating" der Collection an, und speichern diese URI in der Variablen "tableUri". In Zeile 25 rufen wir dann die Funktion "httpGetReq()" auf und übergeben ihr als Parameter die URI. Die Antwort, die wir erhalten ist JSON, so dass wir in Zeile 26 die Funktion "JSON.parse()" benutzen können, um die Antwort für die weitere Verarbeitung in ein Array von JavaScript Objekten zu transformieren. Als erstes bestimmen wir, wie viele Objekte wir erhalten haben. Dafür benutzen wir in Zeile 27 die Methode "length" und speichern die Zahl in der Variablen "count".

```
 24        var tableUri = baseUri + '/rating';
 25        var tableResponse = httpGetReq(tableUri);
 26        var tableResponseObj = JSON.parse(tableResponse);
 27        var count = tableResponseObj.length;
 28
 29        divString += count + ' row(s) retrieved.<p />';
 30        for (i = 0; i < count; i++) {
 31          divString += JSON.stringify(tableResponseObj[i]) + '<p
/>';
 32        }
```

In den Zeilen 29-32 setzen wir die gewünschte Ausgabe in der Variablen "divString" zusammen. Zuerst speichern wir in Zeile 29 die Anzahl der erhaltenen Elemente mit dem Zusatz "row(s) received". Anschliessend arbeiten wir uns in einer Schleife durch die einzelnen Elemente des Array "tableResponseObj" und benutzen in Zeile 31 die Funktion "JSON.stringify()", um für jedes Element einen lesbaren String zur Variablen "divString" hinzuzufügen. Die Array-Elemente sind die individuellen Dokumente der Collection "rating". Daher fügen wir nach jedem Element ein <p>-tag ein, so dass jedes Element in einem eigenen Abschitt erscheinen wird.

Damit haben wir schon viel geschafft, aber einige Details sind noch zu implementieren. Eines davon ist die Definition der Variablen "baseUri". Wie beim Linux Hilfsprogramm "cURL" müssen wir auch hier dem Client (dies ist nun unser Browser) mitteilen, wo er die gewünschte Ressource anfordern soll. Wie die meisten URLs beginnen wir die URI mit "http://". Daran hängen wir die IP-Adresse und Portnummer, wo der JSON Wire Listener erreicht werden kann. Schliesslich fügen wir noch einen '/' und den Namen der Datenbank an:

```
  9     var baseUri = "http://localhost:26352/stores_demo";
```

Das letzte Detail ist recht wichtig. Wir müssen die Funktion "httpGetReq()", die wir in der Funktion "getTableData()" aufrufen, implementieren. Zum Glück ist "httpGetReq()" unkompliziert. Eingabeparameter ist die URI und wir wollen die vom JSON Wire Listener erhaltenen Daten zurückgeben. Dazu benutzen wir ein Objekt "httpXml", das mit dem Aufruf von "XMLHttpRequest()" neu erzeugt wird. Wir benutzen dann die Methoden "open" und "send" dieses Objekts. Die Methode "open" erwartet als ersten Parameter die Aktion, die wir ausführen wollen. Dies ist "GET". Der zweite Parameter ist die URI, die angibt, was wir anfordern wollen. Der dritte Parameter mit dem Wert "false" bestimmt, dass wir die Operationen synchron ausführen wollen. Der folgende Aufruf der Methode "send" schickt die Anfrage an den JSON Wire Listener. Da wir im Aufruf von "open" synchrone Operationen spezifiziert haben, kehrt die Methode "send" erst zurück, nachdem die Antwort auf die Anfrage angekommen ist. Daher wissen wir, dass wir nach der Methode "send" die Antwort vom JSON Wire Listener schon im Element "responseText" des Objekts "httpXml" erhalten haben.
Wir geben also "responseText" an den Aufrufer der Funktion "httpGetReq()" weiter.

```
11      function httpGetReq(Uri) {
12        var httpXml = null;
13
14        httpXml = new XMLHttpRequest();
15        httpXml.open( "GET", Uri, false );
16        httpXml.send( null );
17        return httpXml.responseText;
18      }
```

Nun haben wir allen nötigen JavaScript-Code, um eine "GET"-Anfrage mit der URI für die Collection "rating" an den JSON Wire Listener zu schicken. Wir empfangen die Antwort des JSON Wire Listener und verarbeiten diese mit geringem Aufwand zu einer lesbaren Ausgabe. Die Verarbeitung besteht darin, die Anzahl der JSON-Dokumente zu bestimmen, und eine minimale HTML-Formatierung der Daten vorzunehmen, bevor wir diese dem HTML Division-Element zur Anzeige zuweisen. Nachdem wir das HTML-Dokument, nun mit dem kompletten JavaScript-Code, erneut in den Browser geladen haben, können wir nun auf den "Go"-Knopf klicken, um die Daten von der Collection in der Datenbank zu erhalten.

Die Ausgabe zwischen den zwei horizontalen Linien sollte wie im Abbild 2 aussehen.

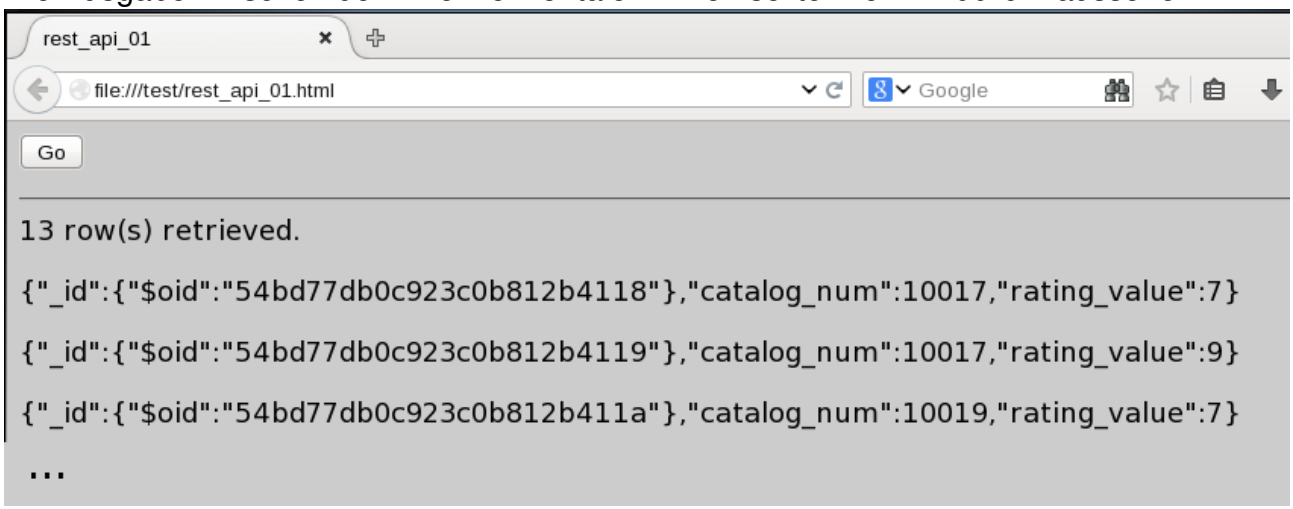


Abb. 2: Das Browserfenster nach dem Klick auf "Go"

Damit der JavaScript-Code einfach und verständlich bleibt, haben wir die empfangenen Daten nur minimal formatiert. Wir haben dies durch Aufruf fertig verfügbarer Funktionen wie "JSON.stringify()" erreicht, mit dem Kompromiss, dass das resultierende Format immer noch JSON-Syntax ist. Trotzdem kann der gezeigte JavaScript-Code als Basisgerüst dienen für eine Verbindung zum JSON Wire Listener und den Empfang seiner Daten.

Komplexere Funktionen zur besseren Formatierung der Daten können darauf aufbauen, ganz nach individuellen Wünschen bezüglich der Darstellung der Daten.

Schliesslich, zur Referenz, das komplette HTML-Dokument mit dem eingebetteten JavaScript-Code:

```
 1  <html>
 2  <head>
 3  <title>rest_api_01</title>
 4  </head>
 5  <body>
 6
 7  <script language="JavaScript">
 8
 9    var baseUri = "http://localhost:26352/stores_demo";
10
11    function httpGetReq(Uri) {
12      var httpXml = null;
13
14      httpXml = new XMLHttpRequest();
15      httpXml.open( "GET", Uri, false );
```

```
      httpXml.send( null );
      return httpXml.responseText;
    }

    function getTableData() {
      var i = 0;
      var divString = '';

      var tableUri = baseUri + '/rating';
      var tableResponse = httpGetReq(tableUri);
      var tableResponseObj = JSON.parse(tableResponse);
      var count = tableResponseObj.length;

      divString += count + ' row(s) retrieved.<p />';
      for (i = 0; i < count; i++) {
        divString += JSON.stringify(tableResponseObj[i])
+ '<p />';
      }

      document.getElementById('div_data01').innerHTML =
divString;
    }

</script>

<form>
  <input type="button" value="Go"
onClick="getTableData();">
</form>

<hr />
<div id="div_data01">

</div>
<hr />

</body>
</html>
```

In einer der nächsten Ausgaben des Newsletters nehmen wir nocheinmal Abstand von JavaScript und dem REST API, um zu sehen, wie wir einen Index auf eine NoSQL Collection anlegen können. Wie auch für SQL Tabellen dient ein Index nicht nur dem schnelleren Zugriff auf die Daten, sondern kann auch die Eindeutigkeit von Werten garantieren.

Termine: Informix Technologie Tag 2015

IBM veranstaltet am **Dienstag, den 5.5.2015** in München einen Informix Technologie Tag, bei dem die Gelegenheit besteht, sich über neue Entwicklungen zu Informix und Best Practices für den erfolgreichen
Einsatz von Informix zu informieren. Darüber hinaus besteht auch die Möglichkeit des Erfahrungsaustauschs mit IBM Mitarbeitern u.a. aus dem Bereichen Entwicklung und Services.

Das Ziel dieser Veranstaltung ist, Informationen zu drei verschiedenen Themenblöcken zu bieten:

1. Es soll ein Überblick über die neuen Entwicklungen des Informix Servers und des zugehörigen Toolings gegeben werden.
2. Anwender sollen im Rahmen von "Best Practice" Vorträgen informiert werden, was bewährte Lösungen für immer wieder auftauchende Probleme sind.
3. Es soll erläutert werden, wie man die Angebote des Informix Supports und von Services am besten und einfachsten nutzen kann.

Agenda Informix Technologie Tage 5.5.2015 in München

Beginn Thema (Sprecher)

09:00:00 Begrüßung (Markus Rummler)
09:10:00 What's New in Informix? (Scott Pickett)
10:10:00 Vormittagspause
10:30:00 Vorteile des Accelerated Value Programs (Michael Brenninger / Edgar Riegger)
10:50:00 Erfahrungsbericht zum Einsatz der IBM Services bei Reiff in Reutlingen (Tim Schöllhammer / Gerd Kaluzinski)
11:10:00 Wie funktioniert der Informix Support? (Markus Rummler)
11:30:00 Warum will der Support immer Daten von mir, die ich nicht habe? (Elisabeth Bach)
12:05:00 Mittagspause
12:50:00 Fragement Level Statistics (Holger Kirstein)
13:25:00 Fine new onstats explained (Andreas Legner)
14:00:00 Chunk oder DBSpace down –Ursachen und Lösungen (Anait Khatchikian)
14:35:00 Nachmittagspause
14:55:00 Analyse von AF Files und Shared Memory Dumps - Eine Support Perspektive (Andreas Dworsky)
15:30:00 Performance Best Practices (Markus Holzbauer)
16:05:00 16:40:00 00:35 Timeseries
Hedwig Fuchs
16:40:00 17:00:00 00:20 Fragerunde Experten Panel
17:00:00 Ende der Veranstaltung

Termine: 66. IUG Workshop in München

Am **Mittwoch, den 06.05.2015** findet in München der 66. IUG Workshop statt. Die IUG Mitglieder sind wie immer am Vorabend zum berühmten IUG Stammtisch eingeladen.

Das Thema des Workshops ist: **Cloud-Technologie**
Der Link zur Agenda ist unter:
http://www.iug.de/index.php?option=com_content&task=view&id=301&Itemid=394
Die Anmeldung ist unter:
http://iug.bm-evento.de/index.php?option=com_seminar&task=3&cid=31
zu finden.
Der Preis für den Workshop beträgt für **Nichtmitglieder € 162,00 (Netto)**.
Für Mitglieder ist diese Veranstaltung kostenlos.
Der IUG-Stammtisch ist nur für Mitglieder vorgesehen.
Wer überlegt der IUG beizutreten, erhält Informationen unter:
Informix User Group e.V.
Tempowerkring 1a
21079 Hamburg
Fon: 040 / 63 90 19 48 / Fax: 040 / 63 90 19 30
E-Mail: **info@iug.de**

Freizeit Tipp: Apfelblüte am Bodensee

Derzeit stehen rund um den Bodensee die Apfelbäume in voller Blüte. Um dies zu geniessen ist eine Wanderung oder Radtour der beste Weg. Besonders zu empfehlen ist die Strecke von Lindau nach Kressbronn, wobei das Ziel dann ein Biergarten mitten in den Obstplantagen mit Blick auf den See sein könnte.

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 (Bodensee bei Friedrichshafen)
Gerd Kaluzinski (Apfelbaum im Redaktionsgarten)