APEX? Es muss nicht immer die Oracle DB sein!

Jürgen Schuster
IT-Beratung
München

Schlüsselworte

Web 2.0, REST, Web Services, APEX, SAP

Einleitung

APEX ist toll, aber was, wenn die Daten nicht in der Oracle DB liegen? An einem konkreten Projektbeispiel wird demonstriert, wie man APEX als Entwicklungsumgebung weiter nutzen kann, obwohl die Daten in einer SAP HANA liegen und in Echtzeit verarbeitet werden sollen. Über REST Webservices lassen sich beliebige Daten aus beliebigen Quellen in APEX verarbeiten. Lernen Sie die verschiedenen Architekturansätze und die zur Zeit beste Lösung für diese Anforderung.

Der Projekthintergrund

Web-Services sind grad in aller Munde. Aber was interessiert es uns als APEX Entwickler? Für uns ist die Welt doch einfach. Wir stellen einen Request an die Oracle Datenbank, unsere Daten liegen in der Oracle Datenbank, das APEX Repository liegt in der Oracle Datenbank, die APEX Engine liegt in der Oracle Datenbank, unsere Businesslogik, die wir in PL/SQL und SQL entwickelt haben, liegt in? Genau in der Oracle Datenbank. Also wo brauchen wir hier Webservices?

Auch bei einem Marktanteil von 75% kommt es vor, dass die Daten nicht in einer Oracle DB liegen. So wie in dem vorgestellten Projektbeispiel, bei dem die Daten in einer SAP HANA Datenbank liegen und auch bleiben sollen.

In der Regel werden Projekte mit der Datenbank eigenen Technologie durchgeführt. Un man hat auch in der Tat zunächst versucht, das Projekt mit dem HANA eigenen Entwicklungssystem SAP UI5 durchzuführen. Bei SAP UI5 handelt es sich um ein JavaScript basiertes Framework, ähnlich wie Ext-JS. Dieses läuft auf dem HANA Dabenbankserver in einem Applicationserver, genannt XS für “Extra Small”. SAP hat also erkannt, dass ein Application Server einen erheblichen Overhead sowohl bei der Softwareentwicklung als auch für den Betrieb darstellt. Dies war der Grund den Application Server direkt in den Datenbankserver zu integrieren, um zumindest den Betrieb eines zusätzlich Servers zu vermeiden.

Ein weiterer cleverer Schachzug war es, nicht Java als Programmiersprache für den Application Server zu wählen, sondern JavaScript. Dazu wurde die JavaScript Engine Spidermonkey von Mozilla verwendet. Damit lässt sich jetzt also auch Business-Logik in JavaScript anstatt Java entwickeln.

Vorteil ist, dass man davon ausgehen kann, dass Web-Entwickler JavaScript bereits können. Außerdem ist Javascript einfacher zu erlernen als das strikte objektorientierte Konzept von Java. JavaScript auf Serverseite ist auch durch Node.js sehr populäre geworden. Auch Oracle bietet nun einen Datenbank-

treiber für Node.js. Das ist für mich ein bisschen wie ein Ritterschlag, sonst würden Oracle da nichts investieren.

Dass JavaScript für Applikationsentwicklung eine gute Idee ist, darauf ist jetzt auch Oracle gekommen, die immerhin Java gekauft und ihre neue E-Business-Suite komplett in Java entwickelt haben. Auf der Oracle Open World 2015 wurde jetzt JET vorgestellt, ein JavaScript Framework, als Erweiterung für die E-Business-Suite. JavaScript ist also voll im Trend. Wobei das Wort nicht wirklich passt, ich selbst verwende JavaScript im Webumfeld seit fast 20 Jahren und ich glaube das wird sich noch weitere 20 halten.

Die Frage, die noch offen bleibt ist, wie mit SAP UI5 auf die Daten in der HANA Datenbank zugegriffen wird? Über REST Webservices! Ich weiß, ein langer Umweg, aber vertraut mir, es ist notwendig die ganze Geschichte zu verstehen, warum ich der Meinung bin, dass APEX sich nicht mehr nur auf die Verarbeitung der Daten in einer Oracle Datenbank beschränken sollte. Und das ist auch die Meinung des Gründers von APEX Mike Hichwa, dass er sich APEX als Entwicklungssystem für alle Datenquellen wünschen würde, die via REST erreichbar sind.(zu hören in der ersten APEX Podcast Episode apex.press/talkshow.

Stellt euch vor, es gäbe kein APEX…

Wir gesagt, haben das einem Mann zu verdanken, der auf die geniale Idee gekommen ist, Web-Entwicklung repositorybasiert direkt in der Datenbank zu betreiben (die sogn. Engine) und dafür eine Web-Anwendung zu entwickeln, um dieses Repository web-basiert zu befüllen (die erste APEX Applikation).

APEX wurde 5 Jahre nur Oracle intern verwendet und es wäre wohl nie zu einem Produkt geworden, wenn nicht Safra Catz (heute CEO von Oracle) das Potential von APEX nicht erkannt und dafür gesorgt hätte, das APEX als No Cost Option mit der Datenbank ausgeliefert wird.

Also APEX war nie als Produkt geplant und wurde nicht im Product Development entwickelt. Vielleicht fällt es jetzt etwas leichter, sich vorzustellen, dass APEX nicht existieren würde…

Womit würde ich dann heute Web-Applications mit Datenbankanbindung entwickeln? Mit Sicherheit mit einem JavaScript Framework, wie Ext-JS, SAP UI5 oder vielleicht jetzt mit JET. Wie würde dann mein Arbeitstag aussehen? Ich würde pro Webseite ca. 5.000 Zeilen Javascript Code schreiben und ebenfalls viel Code für die Controller-Logik. Damit hab ich dann aber nur das Frontend entwickelt.

Ich weiß, es ist kaum zu glauben, dass wir das in APEX gar nicht kennen, da all dies bereits von APEX für uns generiert wird, ohne, dass wir eine Zeile Code schreiben müssen und ohne, dass auch für Frontend und Controller-Logik eine Zeile Code generiert wird.

Ganz zu schweigen, dass es in der Natur der JavaScript Frameworks liegt, dass ihr User Interface mehr oder weniger immer gleich aussieht, da dieses erst am Browser durch das JavaScript generiert wird. Mit APEX hingegen, lässt sich jedes erdenkliche Frontend generieren.

Das ist die Situation. Sobald man also einmal ein Projekt mit APEX gemacht hat, fällt es schwer wieder in die klassischen Entwicklungsansätze zurückzukehren und für ein schlechteres Ergebnis 20 Mal mehr Entwicklungszeit zu investieren, ganz zu schweigen von den Maintenance Kosten, denn jede Zeile Sourcecode eines Projektes kostet jeden Monat Geld für Wartung, Pflege und Change Requests.

So kam man auch bei meinem Kunden auf die Idee, das Projekt in APEX durchzuführen, obwohl die Daten in SAP HANA liegen und dort auch bleiben sollen, um Echtzeitauswertungen durchführen zu können.

Das angebotene Gateway war, neben hohen Lizenzkosten, fehlerhaft. Also wenn SAP UI5 selbst auf die eigene Datenbank via REST Webservice zugreift, können wir das nicht auch mit APEX? Um das zu klären, ist es zunächst wichtig Web-Services zu verstehen.

Webersvices

Wie kriegen wir denn die Daten in den Browser, z. B. mit APEX? Der Browser Request geht erst mal per HTTP an den Webserver. Der kann aber mit der Anfrage gar nichts anfangen. Es gibt aber sogenannte Module, die dem Webserver Spezialaufgaben abnehmen, z. B. Daten aus einer Datenbank zu holen. Für die Oracle Datenbank gibt es für den Apache Webserver das mod_plsql, geschrieben in C von Tom Kyte (asktom.oracle.com). Dieses Modul ist wie ein kleines SQL-Plus, welches über OCI eine Datenbankverbindung öffnet und auf den dort generierten HTML Code wartet, bevor das Ergebnis über den Webserver zurück an den Browser geliefert wird, wobei danach sofort die Datenbankverbindung wieder gekappt wird.

Also bekommt der Browser über HTTP HTML-Code. Bei sogn. AJAX Requests kommen auch manchmal nur Daten zurück, ohne dass sie in HTML verpackt sind und das meistens im JSON Format, ein Format, das JavaScript nativ verstehen kann, in etwa wie ein Cursor für in PL/SQL

Das ist ein wichtiger Denkschritt. Wir kriegen Daten im JSON Format über einen HTTP Request. Die Daten kommen noch aus der Oracle Datenbank, aber spätestens der Webserver weiß das nicht mehr.

Das eröffnet nun ganz andere Möglichkeiten. Wenn es spätestens dem Webserver egal ist, wo die Daten herkommen, warum kann man dann nicht die Daten aus einer anderen Datenquelle über HTTP holen, wie bei einem AJAX Request?

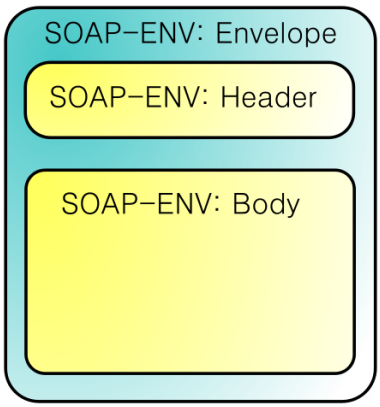
Um aber eine APEX Webseite aufzubauen, müssen wir zunächst mal in die Datenbank. Dort spielt sich alles ab. Aber auch die Datenbank kann, wie ein Browser, einen HTTP Reqeust abschicken zu einem Webserver, um über diesen von einer anderen Datenquelle Daten bekommen. Diese Daten können dann in der Oracle Datenbank verarbeitet werden (z. B. in eine temporäre Tabelle speichern, in APEX haben wie dazu das Konstrukt einer Collection) und schon sind wir wieder in der Oracle Welt und können mit den Standardfunktionlitäten von APEX arbeiten, als wenn die Daten originär in der Oracle Datenbank gespeichert worden wären.

SOAP und REST

Fast müsste ich sagen, es gab zwei Arten von Webservices. SOAP und REST. In der Praxis finden wir, wenn es sich nicht um Altanwendungen handelt, nämlich nur noch REST.

SOAP:

Obwohl SOAP Simple Object Access Protocol heißt, ist es - im Gegensatz zu REST - gar nicht so simple. Um vom Browser einen SOAP Request zu schicken musste dieser Request in einen sogn. Umschlag verpackt werden. In dem Umschlag gibt es dann die Nachricht, die wieder einen Header haben kann auf jeden Fall braucht es einen Body.



Auch die Antwort kommt in einem “Brief”. Das ganze basiert auf XML und man muss sowohl für die Anfrage als auch für die Antort die Strukturen des Dienstes genau kennen, damit das funktioniert. Hier ein Beispiel einer Mini-Antwort. Man bekommt ein Gefühl, wie kompliziert das bei komplexeren Reqeust / Anwtorten werden kann und wieviel “Verpackung” da übers Netz geht, was anschließend über einen Parser wieder ausgepackt werden muss.

```
<?xml version="1.0"?>
<s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope">
    <s:Header>
        <m:RequestID xmlns:m="http://www.lecture-db.de/soap">a3f5c109b</m:RequestID>
    </s:Header>
    <s:Body>
        <m:DbResponse xmlns:m="http://www.lecture-db.de/soap">
            <m:title value="DOM, SAX und SOAP">
                <m:Choice value="1">Arbeitsbericht Informatik</m:Choice>
                <m:Choice value="2">Seminar XML und Datenbanken</m:Choice>
            </m:title>
        </m:DbResponse>
    </s:Body>
</s:Envelope>
```

Dann muss außerdem noch beschrieben werden, wie so ein SOAP Service aussieht und wie er aufgerufen werden kann, mit welchen Parametern, was zurückkommt usw.. Dafür gibt es dann eine WSDL (Web Services Description Language). Das muss vom Anbieter generiert werden. Also alles in allem nicht so Simple und vor allem viel Verpackung:

```
<definitions name="StockQuote"
        targetNamespace="http://example.com/stockquote.wsdl"
        xmlns:tns="http://example.com/stockquote.wsdl"
        xmlns:xsd1="http://example.com/stockquote.xsd"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
     <schema targetNamespace="http://example.com/stockquote.xsd"
          xmlns="http://www.w3.org/2001/XMLSchema">
        <element name="TradePriceRequest">
           <complexType>
               <all>
                   <element name="tickerSymbol" type="string"/>
               </all>
           </complexType>
        </element>
        <element name="TradePrice">
           <complexType>
               <all>
                   <element name="price" type="float"/>
               </all>
           </complexType>
        </element>
     </schema>
  </types>

  <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
  </message>

  <message name="GetLastTradePriceOutput">
      <part name="body" element="xsd1:TradePrice"/>
  </message>

  <portType name="StockQuotePortType">
      <operation name="GetLastTradePrice">
         <input message="tns:GetLastTradePriceInput"/>
         <output message="tns:GetLastTradePriceOutput"/>
      </operation>
  </portType>

  <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetLastTradePrice">
         <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
         <input>
             <soap:body use="literal"/>
         </input>
         <output>
             <soap:body use="literal"/>
         </output>
      </operation>
  </binding>

  <service name="StockQuoteService">
      <documentation>My first service</documentation>
      <port name="StockQuotePort" binding="tns:StockQuoteSoapBinding">
         <soap:address location="http://example.com/stockquote"/>
      </port>
  </service>

</definitions>
```

REST:

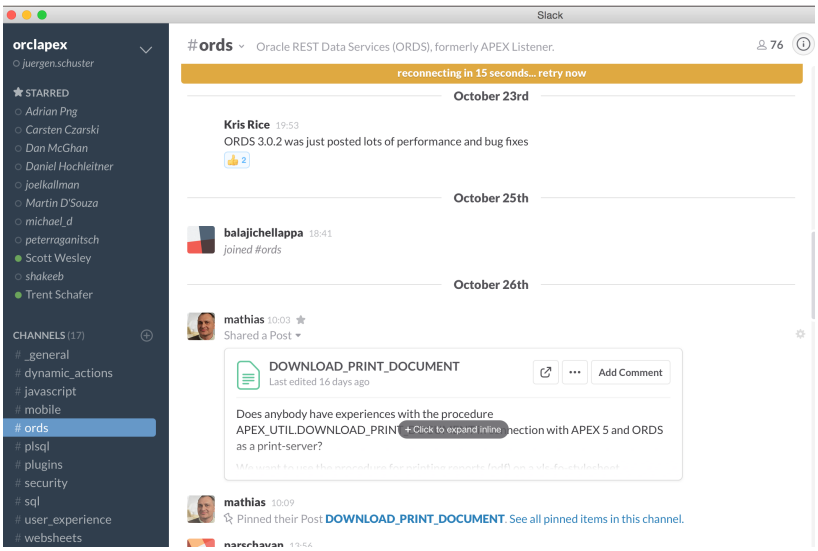
Wenn man einmal REST gesehen hat, kann man gar nicht verstehen, wie man jemals auf SOAP gekommen ist. Das eine war wohl eher ein universitärer Ansatz, REST kommt sicher von Leuten, die damit auch selbst arbeiten wollten ;-)

Hier ein Beispiel um alle Datensätze aus der emp Tabelle zu holen:

https://apex.world/ords/emp

Mehr ist es nicht, da kommen dann noch mal hin und wieder Parameter an die URL oder man verschickt einen Request per POST, was am Ende aber nicht komplizierter ist.

Damit habt ihr die Basis, um den Vortrag noch besser zu verstehen :-) Wenn ihr Fragen dazu habt, könnt ihr euch gerne melden. Es gibt auch einen extra Slack channel in dem auch Kris Rice aktiv ist und alle Tech-Pro's. Man kommt da über apex.world rein.



Kontaktadresse:
Jürgen Schuster
IT-Beratung
Camerloherstr. 64
80689 München

Telefon: +49 170-4759357
Fax: +49 89 560 16 147
E-Mail: j_schuster@me.com
Twitter: @JuergenSchuster
Internet: http://juergen-schuster-it.de
APEX Podcast: http://apex.press/talkshow
Projekt: http://apex.world