Das richtige Tool – ein Plädoyer für Apex

Denes Kubicek

Als der Autor Ende 2004 mit Apex angefangen hat, war das für ihn eine wirkliche Entdeckung. In der Zeit davor war alles viel zu umständlich. Applikationen konnte man nur mithilfe von Forms erstellen und das hat eine richtige Infrastruktur erfordert. Man musste einen Application Server installieren. Auf der Client-Seite waren Applets erforderlich und für das einfache Berichten musste man sich mit Oracle Discoverer oder mit Reports auskennen.

Apex hat neue Möglichkeiten aufgezeigt. Ganz neu war Apex jedoch nicht. Der Vorgänger „WEB DB" hat schon gezeigt, in welche Richtung es mit Anwendungsentwicklung in Zukunft gehen wird: Entwicklung und Deployment ausschließlich im Browser; keine aufwändige Installation und keine Add-ons. Schlank und schnell. Leider wurde die Weiterentwicklung von WEB DB im Jahr 2000 gestoppt und danach herrschte einige Jahre Stille. Mit der Version 1.4 lebte Apex wieder auf und begann seinen Platz in der Oracle-Community einzunehmen.

Wie bei allen anderen Sachen gibt es auch bei Apex zwei Lager – Befürworter und Skeptiker. Die Befürworter waren immer damit beschäftigt, die guten Eigenschaften von Apex in den Vordergrund zu stellen. Im Fokus stand immer das, was Apex kann und eines Tages können wird. Die Skeptiker dagegen beschäftigten sich mit dem, was Apex nicht kann. Dadurch sind viele Vorurteile entstanden, mit denen man auch heute noch kämpfen muss.

Vorurteile und Fakten

In Bezug auf Apex gibt es einige Vorurteile. Diese sind in den Zeiten entstanden, als Apex noch relativ unbekannt war. Auch Oracle selbst wusste das Potenzial von Apex nicht vom Anfang an richtig einzuschätzen. Es war als ein Ersatz für Microsoft Access gedacht und somit unabsichtlich in eine kleine Nische gedrängt. Dadurch ist auch das Vorurteil entstanden, Apex sei nur für einen bestimmten Typ von Anwendungen geeignet. Die Frage des „Geeignet sein" begleitet Apex heute noch. Fragen wie „Ist Apex für das Programmieren der großen Anwendungen geeignet?" sind immer wieder zu hören.

Die Antwort „Ja, durchaus" ist nicht einfach eine pauschale Einschätzung, sondern basiert auf Fakten. Der Autor selbst war an mindestens einem Dutzend solcher Anwendungen beteiligt. Es ging hier unter anderem um drei DAX-Unternehmen, die sich für Apex als Entwicklungsplattform entschieden haben und bei denen er die Gelegenheit hatte, an solchen großen Anwendungen mitzuarbeiten.

Die Befürchtung, die Anwendung gehe in die Knie, wenn sich mehr als eine Handvoll Benutzer einloggt, stammt noch aus den Zeiten von Microsoft Access. Dieses Programm war nicht unbedingt für ein Netzwerk gedacht und hatte genau diese Probleme. Der Vergleich mit Access hat dazu geführt, dass diese Zweifel auch bei Apex aufkommen. Apex kennt jedoch keine Beschränkung der Benutzerzahl, höchstens ein Limit in der Default-Einstellung des aktuellen Webserver-Dienstes. Apex selbst kommt mit einer großen Anzahl von Benutzern klar. Der beste Beweis dafür ist auf *„https://apex.oracle.com"* zu finden. Dort wird ein kleiner Server mit bescheidenen Ressourcen eingesetzt, um mehr als 12.000 Workspaces und etwa sechs Millionen Seitenaufrufe im Monat zu bedienen.

Auch die Frage „Wird Apex von Oracle unterstützt?" lässt sich mit „Ja" beantworten. Apex ist ein Datenbank-Feature und wird wie alle anderen Bausteine voll unterstützt. Es wird regelmäßig gewartet und mit Patches beliefert; die Updates erscheinen in der Regel alle eineinhalb Jahre. Alle drei Jahre gibt es ein Major Release. Nicht ohne Grund steht auf der offiziellen Apex-Seite: „Oracle Application

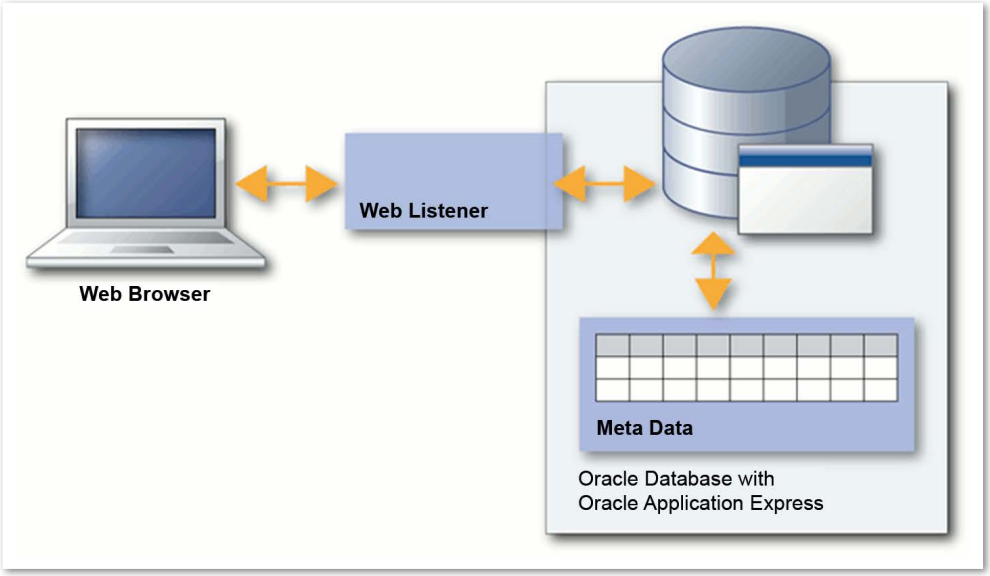

Abbildung 1: Die Apex-Architektur

Express (Apex) is Oracle's primary tool for developing Web applications with SQL and PL/SQL. Using only a web browser, you can develop and deploy professional Web-based applications for desktops and mobile devices."

Dass Apex in Zukunft kostenpflichtig wird, glaubt der Autor nicht. Wobei Apex nicht kostenlos ist; es ist ein Bestandteil der Datenbank und diese kostet Geld. Apex kam als kostenloses Feature dazu. Es ist höchst unwahrscheinlich, dass Apex eines Tages von Oracle separat berechnet wird.

Mancher Entwickler stellt auch die Frage: „Kann ich Apex in Subversion einchecken und jederzeit wieder auf den alten Stand bringen?" Das geht, aber wozu? Es geschieht sehr häufig, dass Apex mit Entwicklungsumgebungen verglichen wird, die aus dem Java-Umfeld stammen. Sehr häufig wird Apex auf dieser Basis beurteilt – funktioniert es denn genauso wie bei Java? Falls nicht, dann ist das keine gute Idee. Diese Einstellung beruht auf dem mangelnden Wissen über das Konzept, die Architektur und die Technologie. Apex ist kein externes Tool, sondern Bestandteil der Datenbank, das dort verwaltet und gesichert wird. Apex wird zentral und nicht dezentral entwickelt. Das ist seine Stärke und natürlich, wenn man das dezentrale Konzept mag, auch seine Schwäche.

Eine Technologie, die begeistert

Das Konzept von Apex ist sehr einfach und begeistert deswegen umso mehr. Apex ist Bestandteil der Datenbank und wird dort verwaltet. Wenn man keinen weiteren Webserver-Dienst installieren möchte, übernimmt die Datenbank mit dem „Embedded PL/SQL Gateway" auch diese Rolle. Das Ganze ist sehr schnell installiert und sofort startbereit. Man kann nur mithilfe eines Browsers die ganze Arbeit erledigen. Auch die Benutzung einer Anwendung geschieht im Browser *(siehe Abbildung 1)*. Es sind keine weiteren Tools und Add-ons zwingend notwendig – im Gegensatz dazu, was man aus den anderen Frameworks kennt.

Für diejenigen, die keine Vollversion der Oracle-Datenbank haben, steht die kostenlose Version XE11 zur Verfügung *(Download siehe „http://www.oracle.com/technetwork/database/database-technologies/express-edition/downloads/index.html")*. Diese Version lässt sich ohne zusätzliche Konfiguration auf jedem Rechner installieren und bietet einen Schnellstart in die Apex-Welt. Nachdem die Software installiert wurde, genügt ein Link-Aufruf im Browser und man steckt sofort mittendrin.

Vor nicht allzu langer Zeit hat der Autor ein Artikel darüber gelesen, für wen Apex als Entwicklungsplattform geeignet ist. Man spottete über Apex und sagte, es sei am besten für sogenannte „Halbwissende" geeignet und die meisten, die sich damit beschäftigen, gehörten zu dieser Gruppe. Es mag sein, dass dem so ist. Das ist aber auch das Schönste an Apex. Es erfordert keine langjährige Studienzeit, bevor man überhaupt eine Seite in einer Anwendung erstellen kann. Dem Autor fällt dabei als Beispiel die Anleitung im „Oracle Magazine" zur Erstellung der Wertelisten für ADF ein. Es waren etwa vier komplette Seiten an Text notwendig, um zu erklären, wie man eine Werteliste erstellen und in den Anwendungen bereitstellen kann. Das Erstellen von Wertelisten in Apex ist dagegen ein Selbstläufer. Wie für die meisten anderen Features braucht man dafür keine separate Anleitung.

Mit Apex kann man seine Vorstellungen sehr schnell verwirklichen. Man kann es jemandem geben, der über kein

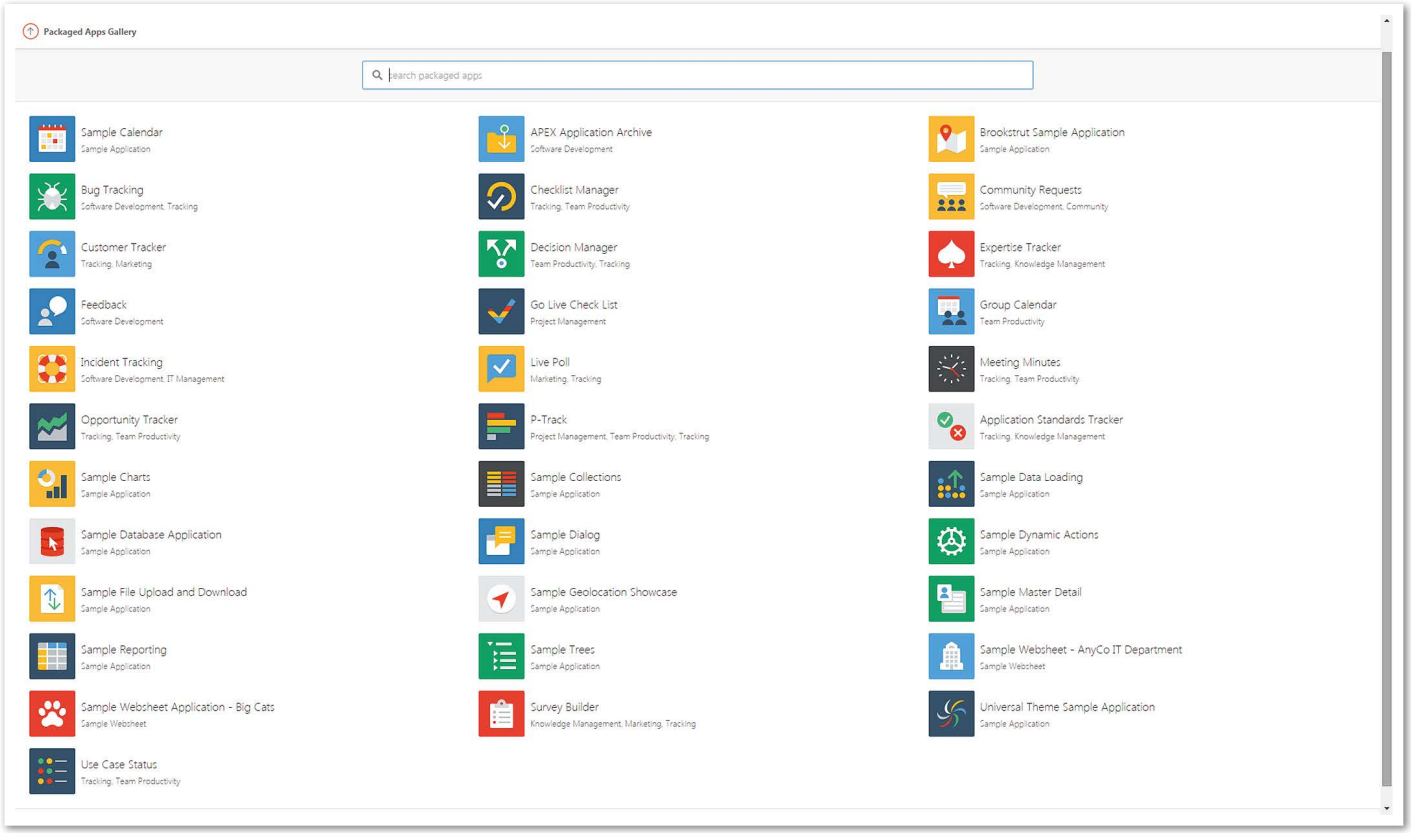

Abbildung 2: Apex Packaged Applications

Oracle-Wissen verfügt, oder jemandem, der schon sehr viel über SQL und PL/SQL weiß. Die Ergebnisse werden gleich sein. Beide können in wenigen Stunden eine brauchbare Anwendung auf die Beine stellen. Apex kann auch als Mockup-Tool sehr gut eingesetzt werden. Wer bereit ist, etwas Neues kennenzulernen, keine Vorurteile hat und nichts von vornherein ausschließt, ist für Apex geeignet und sollte es unbedingt ausprobieren.

Packaged Applications

Packaged Applications sind ein sehr nützliches Apex-Feature. Mit jeder neuen Version wird es verbessert und ausgebaut. Packaged Applications sind verschiedene Anwendungen, die sofort und uneingeschränkt zur Verfügung stehen. Mit einigen Klicks kann man eine Anwendung installieren. Dabei werden auch alle dafür notwendigen Objekte und Daten installiert *(siehe Abbildung 2)*.

Der Code und die Lösung eines Problems können direkt betrachtet und komplett oder zum Teil wiederverwendet werden. Die Anwendungen sind thematisch unterschiedlich und decken eine ganze Reihe verschiedener Lösungen ab. In der derzeitigen Version 4.2.0.6 sind es insgesamt vierzig Anwendungen. In der neuen Version 5.0 wird das konsolidiert und es kommen neue Themen dazu.

Die wichtigsten Apex-Features

Oracle Apex stellt alles bereit, was man für eine gute Web-Anwendung benötigt. Die wichtigsten Features sind:

- Entwicklung und Deployment von Desktop- und mobilen Anwendungen ausschließlich im Browser
- jQuery-Support als Standard
- Die Möglichkeit, alle Datenbank-Features in Apex einzusetzen (Spatial, Oracle Text, File Handling, Emailing, XML, JSON etc.)
- Apex ist genauso sicher wie die Datenbank selbst
- Authentifizierungs- und Autorisierungsmöglichkeiten können an alle beliebigen Systeme angebunden werden (SSO, LDAP, Active Directory)
- SQL-Abfragen sind viel schneller als mit anderen Tools, weil sie direkt in der Datenbank ausgeführt und durch Optimizer verwaltet werden
- Plug-ins und Dynamic Actions bieten die Möglichkeit, externe Ressourcen (JavaScript Libraries) so in die Anwendung einzubinden, als wären sie eine Standard-Komponente
- Interactive Reports sind eine sehr gute Möglichkeit, Excel-ähnliche Mittel bereitzustellen, mit denen Benutzer die Datenbank-Tabellen nach dem Excel-Prinzip abfragen können. Die neuen Interactive Reports in der Version 5.0 bieten auch die Erstellung von Pivot-Ansichten im Standard
- Die Erstellung von Charts „on the fly" ist als Standard-Komponente in einem Interactive Report enthalten
- Die Bereitstellung von RESTful-Webservice ist in der Kombination mit den Oracle-Rest-Data-Services möglich *(sie-*

Oracle verlängert Support für Forms und Reports 11g R2

In den aktuellen Support Policies hat Oracle die Verlängerung des Supports von Forms und Reports 11g R2 angekündigt *(siehe „http://www.oracle.com/us/support/library/lifetime-support-middleware-069163.pdf")*. Der „Premier Support" für beide Tools wird demzufolge um zwei weitere Monate bis Dezember 2016 verlängert, der „Extended Support" um 14 Monate bis Dezember 2018 ausgeweitet. Die Version 11g R1 ist davon nicht betroffen *(siehe Tabelle)*.

Die Ankündigung des verlängerten Supports könnte wie schon in der Vergangenheit bei früheren Versionen ein Hinweis auf das baldige Erscheinen von Forms und Reports 12 sein. Michael Ferrante, Oracle Product Manager Forms, hatte bereits auf der DOAG 2014 Konferenz + Ausstellung das Erscheinen von Version 12 von Forms und Reports im Kalenderjahr 2015 angekündigt. Damit widerlegte er gleichzeitig auch unbestätigte Informationen darüber, dass Oracle Reports nicht in Version 12 erscheinen könnte.

Ferrante legte außerdem auf Twitter allen Kunden, die noch Oracle Forms 6 nutzen, ein Upgrade auf Version 11 nahe. Oracle stellt hierfür einen Upgrade Guide zur Verfügung *(siehe „http://docs.oracle.com/cd/E48391_01")*.

Release	GA Date	Premier Support Ends	Extended Support Ends	Sustaining Support Ends
Portal 11gR1, Forms 11gR1, Reports 11g R1 and Discoverer 11g R1 (11.1.1.x)	Jun 2009	Jun 2014	Jun 2017	Indefinite
Forms and Reports 11g R2 (11.1.2.x)	Oct 2011	Dec 2016	Dec 2018	Indefinite

Oracle Fusion Middleware Releases

he „http://www.oracle.com/technetwork/developer-tools/rest-data-services/overview/index.html“)

- Native Excel-Upload ist ebenfalls in Kombination mit den Oracle-Rest-Data-Services möglich. Durch eine einfache Erweiterung der Parameter lassen sich Excel-Dateien im Original-Format hochladen und die Daten einlesen. Auch das Einlesen von beliebig vielen Datenblättern funktioniert problemlos
- Apex eignet sich für die Entwicklung und Bereitstellung von multisprachigen Anwendungen. In der neuen Version 4.2.0.6 ist es nun möglich, die Anwendungen auch ohne neues Deployment aus der Benutzeroberfläche der primären Anwendung heraus zu übersetzen
- Apex ist eine offene Plattform. Man kann den Code nicht verstecken. So steht für alles, was man in Apex einsetzen kann, auch das dazugehörige API zur Verfügung. Dadurch lässt sich alles, was man in der Apex-Oberfläche erledigen kann, genauso im Backend anstoßen

Diese Liste ist nicht vollständig und ein Teil ist sicherlich subjektiv empfunden. Sie zeigt aber, dass Apex durchaus eine sehr ernsthafte Alternative zu anderen älteren Frameworks geworden ist.

```
BEGIN
   package_name.funktion_oder_procedure (:p1_variable_1,
                                         :p1_variable_2,
                                         :p1_return_wert
                                         );
END;
```

Listing 1

```
SELECT column1, column2, column3
      FROM my_table
   WHERE column_2 = :p1_variable_1
```

Listing 2

Community und Ressourcen

Eines der großartigen Dinge bei Apex ist die herausragende Community. Apex bedeutet Web – es wird im Web gelernt, gelebt und weiterentwickelt. Wenn man eine Lösung für sein Problem benötigt, ist das Web die Stelle, an die man sich wendet. Das Know-how wird gerne weitergereicht und so etwas wie Geheimhaltung ist dort völlig unbekannt.

Nachfolgend sind einige der wichtigsten Ressourcen im Web aufgelistet, allesamt ein Muss. Sie beinhalten schon alles, was man über Apex wissen muss. Von dort kommt man dann auch zu den anderen Ressourcen, für die letzten Details:

- *Oracle Apex Forum*
  https://community.oracle.com/community/database/developer-tools/application_express

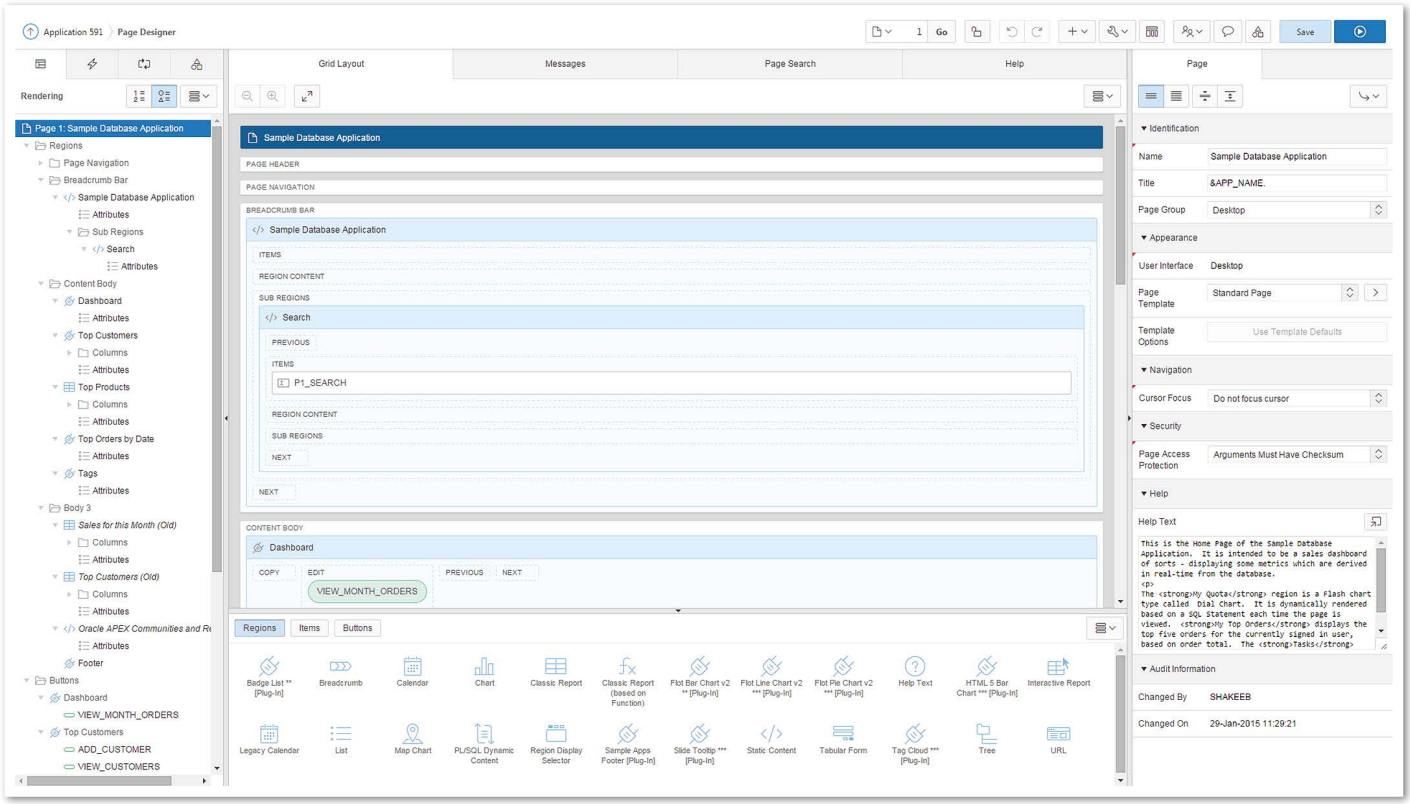

Abbildung 3: Der neue Page-Designer

- *Deutsche Oracle Apex Community*
  http://www.oracle.com/webfolder/technetwork/de/community/apex/index.html
- *All Apex Blogs*
  http://www.odtug.com/apex
- *Oracle Apex Documentation*
  http://www.oracle.com/technetwork/developer-tools/apex/documentation/index.html
- *Oracle Apex Workspace*
  https://apex.oracle.com/pls/apex/f?p=4550:1
- *jQuery Ressourcen*
  http://stackoverflow.com

Zu beachten

Apex ist kein Wundermittel für das Programmieren von guten Anwendungen. Nur deswegen, weil eine Anwendung mit Apex umgesetzt wurde, wird sie nicht automatisch gut. Wenn man mit Apex komplexere Anwendungen programmieren möchte, müssen zwei Voraussetzungen erfüllt sind:

- Sehr gute und fundierte SQL- und PL/SQL-Kenntnisse
- Sofern das Layout und die Ergonomie eine größere Rolle spielen, sind Kenntnisse in jQuery, CSS und HTML5 unausweichlich

In Apex gibt es viele Stellen, die Code aufnehmen können. Doch nur deshalb sollte man das nicht machen. Wenn keine Planung und Best-Practices vorliegen, wird dies zur sogenannten „Code-Redundanz" führen. In der Praxis kommt es sehr häufig vor, dass jeder Entwickler für sich mit den anonymen PL/SQL-Blöcken arbeitet. Wenn eine komplexe Seite auf diese Weise entsteht, wird sie zwangsweise überladen und mit der Zeit langsam. Die Auswirkungen werden dann erst in der Produktion sichtbar, wenn der Code mit einer richtigen Datenmenge arbeitet.

Komplexe Dinge auslagern

Der Autor empfiehlt an dieser Stelle, die Business-Logik in die Packages auszulagern. Der Code sollte an bestimmten Stellen nie zum Einsatz kommen (auch wenn es möglich wäre) und idealerweise nur aus einem einfachen Aufruf bestehen *(siehe Listing 1)*. Was die SQL-Abfragen betrifft, sollten komplexe Abfragen ebenfalls zuerst in die Views ausgelagert werden *(siehe Listing 2)*.

Sämtliche Joins, Funktionsaufrufe und Unterabfragen sollten innerhalb einer darunter liegenden View verborgen sein. Man sollte diese Logik auf die Spitze treiben und immer darauf achten, es beim nächsten Mal noch ein wenig besser zu machen. Die Gründe dafür sind ganz praktisch und einfach:

- Die Code-Redundanz wird vermieden
- Der Code läuft schneller, da er in die Packages und Views ausgelagert ist
- Das Deployment der Anwendung muss nicht so häufig durchgeführt werden, da es oft reicht, nur das Package oder die View anzupassen
- Durch das Vermeiden der Code-Redundanz reduziert sich auch das Debugging, da sich der versteckte Code nicht wiederholt, sondern zentral verwaltet wird

Blick in die Zukunft

Als Befürworter von Apex ist der Autor natürlich sehr an der Zukunft von Apex interessiert. Durch seine Aktivitäten in der Community hatte er die Möglichkeit, sich an der Entwicklung von Apex 5.0 aktiv zu beteiligen. Die neuen Features sind bereits bekannt, darunter sind:

1. Universal Theme
2. Multiple Interactive Reports
3. Ein neuer Page-Designer mit Drag & Drop und einer Arbeitsweise, die an
4. Oracle Forms erinnert *(siehe Abbildung 3)*
5. Static File Upload
6. Modale Dialoge

Man könnte sagen, dass Apex mit der Version 5.0 so richtig erwachsen wird. Wenn mit den Versionen 5.1 und 5.2 dann noch „jQuery Grid" hinzukommt und die Collections auf mehr als 50 Spalten (200) ausgeweitet werden, wird Apex vom jetzigen Standpunkt aus kaum noch Wünsche übriglassen.

Warum Apex das richtige Tool ist

Wer auf einer Oracle-Datenbank arbeitet und seine Prozesse mit SQL und PL/SQL steuert, für den ist Apex ohne Zweifel die richtige und die beste Wahl.

Apex bietet eine einmalige Architektur, die es ermöglicht, sowohl Desktop- als auch mobile Anwendungen aus einer Hand zu

liefern. Zudem ist alles noch sehr deklarativ und standardisiert.

Es ist ganz klar, dass auf dem Markt viele Frameworks zur Auswahl stehen. Apex wird diesen Konkurrenzkampf nicht immer für sich entscheiden können – manchmal auch deswegen, weil die Unternehmens-Entscheidungen häufig nicht mit rationalen Gründen und Argumenten getroffen werden.

Feedback aus der Community

Der Autor bekommt ab und zu E-Mails wie diese: „My Manager decided to cancel Apex from his list of solutions for the application programming and I tried in many ways to convince him that Apex is a true RAD tool and has very good reporting capabilities, etc... However, this doesn't work and his reasons are not clear to me. Probably based on his .NET background. The only thing he tells me all the time is that .Net is object oriented programming and oracle procedure programming. Please help me to find some things to convince him. I like Apex and I want to continue doing that. Any articles or any professional reasons would be a great help. Thank you very much,...“ Seine Antwort darauf ist, sich sofort einen neue Stelle zu suchen, an der man weiter mit Apex arbeiten kann.

Keine gezielte Begründung

Die Gründe für die Argumentation gegen Apex können verschieden sein. Häufig stehen die Architektur-Vorgaben im Vordergrund, die viele potenzielle Kandidaten ausschließen. Zum Beispiel wird als Voraussetzung festgelegt, dass es einen separaten Datenbank- und einen separaten Applikations-Server geben muss und Anwendungen und Business-Logik getrennt von der Datenbank zu verwalten sind.

Eine ausführliche Begründung, warum das so sein muss, gibt es meistens nicht. Es ist aber klar, für welche Technologien das ausgelegt ist und welche Beweggründe dahinterstehen. Dann hilft hier auch keine Argumentation, dass dies nur Geld kostet und keine Vorteile bringt.

Auf der anderen Seite existieren auch Fälle, bei denen die Kosten und die Zeit eine entscheidende Rolle spielen. Es gibt dafür genügend Beispiele aus der Praxis und dort gewinnt Apex meistens mit einem klaren Vorsprung.

Der Autor kann sich an mehrere Situationen wie diese erinnern:

- Eine Anwendung wurde bei einem Kunden mit Java erstellt und musste nun erweitert werden
- Apex hat dort eine untergeordnete Rolle gespielt und nur Zusatzfunktionen zur Verfügung gestellt, die mit Java aus zeitlichen beziehungsweise Kapazitätsgründen nicht erledigt werden konnten. Das Layout war identisch und somit war die Plattform nur noch am Link im Browser erkennbar
- Der Kunde wollte eine relativ komplexe Verwaltung von Benutzern, Rollen und Rechten realisieren und dabei Pflegemasken haben, mit denen er den Zugriff auf seine Anwendungen gezielt steuern kann
- Dabei wurden Kalkulationen der Aufwände für eine Lösung in Java und eine Lösung mit Apex gemacht
- Der Vergleich fiel für Java vernichtend aus. Die Kalkulation des Aufwands für Java bezifferte man mit etwa 250 Mann-Tagen. Die Kalkulation für Apex dagegen lag bei drei Wochen
- Die Entscheidung war ganz klar und die Umsetzung dauerte am Ende etwas weniger als fünfzehn Mann-Tage

In einem anderen Fall musste aus politischen Gründen eine Funktionalität mit JSF umgesetzt werden. Der Kunde hatte schon Apex im Einsatz und für den Zweck wären Interactive Reports die richtige Wahl. Die Lösung hätte auch mit normalen Reports umgesetzt werden können. Die Anforderung war ziemlich trivial:

- Eine Seite beziehungsweise ein Bericht für das Suchen nach bestimmten komplexeren Informationen war auf Basis eines Header-Datensatzes und einiger Detail-Datensätze zu erstellen
- Vorgabe war, dass die Benutzer ihre Berichte jederzeit speichern und wieder abrufen können
- Einige Prozesse für die nachträgliche Änderung der Daten sollten in die Seite eingebunden werden

Der geschätzte Aufwand für die Umsetzung dieser Anforderungen mit Apex lag bei etwa 20 und für JSF bei 75 Mann-Tagen. Die tatsächliche Realisierung mit JSF dauerte dann rund 200 Mann-Tage.

Fazit

Wenn Kosten, Zeit und Transparenz im Vordergrund stehen, hat Apex gute Chancen, sich durchzusetzen. Je häufiger es sich durchsetzt, desto offensichtlicher werden die Vorteile auch für die Entwicklung aller anderen Anwendungen.

Selbstverständlich ist Apex kein Allheilmittel und kann nicht alles am besten lösen. Auch bei Apex trifft die „80:20“-Regel ganz gut zu. In 80 Prozent der Fälle wird Apex eine ernsthafte Alternative, die man auf jeden Fall berücksichtigen sollte. Und das ist schon sehr viel.

Die aufwändigen Application-Server, die bereits vor der Datenabfrage gewaltige Ressourcen beanspruchen, gehören der Vergangenheit an. In naher Zukunft wird sich auch in der Anwendungsentwicklung die „80:20“-Regel durchsetzen – die meisten Anwendungen, die wir benutzen und kennen, werden im Browser laufen. Nichts spricht mehr für Apex als das.

Denes Kubicek
deneskubicek@yahoo.de