# Tune Up Your APEX

**Oliver Lemm**
**MT-AG**
**Ratingen**

**Schlüsselworte**

APEX, Performance, Tuning, Validierung, Debugging

## Einleitung

APEX bietet mittlerweile für jede Webentwicklung einen interessanten Ansatz. Der Einstieg ist schnell gemacht und die ersten Ergebnisse schnell sichtbar. An diesem Punkt können einzelne Masken schnell überladen wirken und jede Menge Funktionalitäten enthalten, die das Laden der Seite verzögern können. Vor allem beim Laden der Seite ist es frustrierend, wenn die Seite mehr als nur eine Sekunde benötigt. Hier ist entscheidend wo und wie man die Schwachstellen identifiziert und wo die Ladezeit verloren geht. Komplexes SQL bzw. PL/SQL, falsche Nutzung von JavaScript oder Dynamic Actions, sowie die falschen Conditions können sich enorm auswirken.

Vor allem beim Interactive Report ist es fatal, wenn die Performance nicht passt und bei jeder Aktion des Benutzers immer wieder Ladezeit spürbar ist.

Auch Querschnittsfunktionen über Application Prozesse oder Validierungen die SQL/PL/SQL enthalten obwohl es nativ über APEX viel schneller geht können entscheidend sein.

## Die Grundlagen

Um APEX richtig und performant zu betreiben ist es maßgeblich für jeden Entwickler zu verstehen wie genau APEX funktioniert. Hier sollte man sich als allererstes vor Augen führen, dass jeder Seitenaufbau dynamisch durchgeführt wird. Es werden also bis auf das Template alle Komponenten, Prozesse und was an SQL oder PL/SQL verwendet wird beim Rendern ausgewertet.

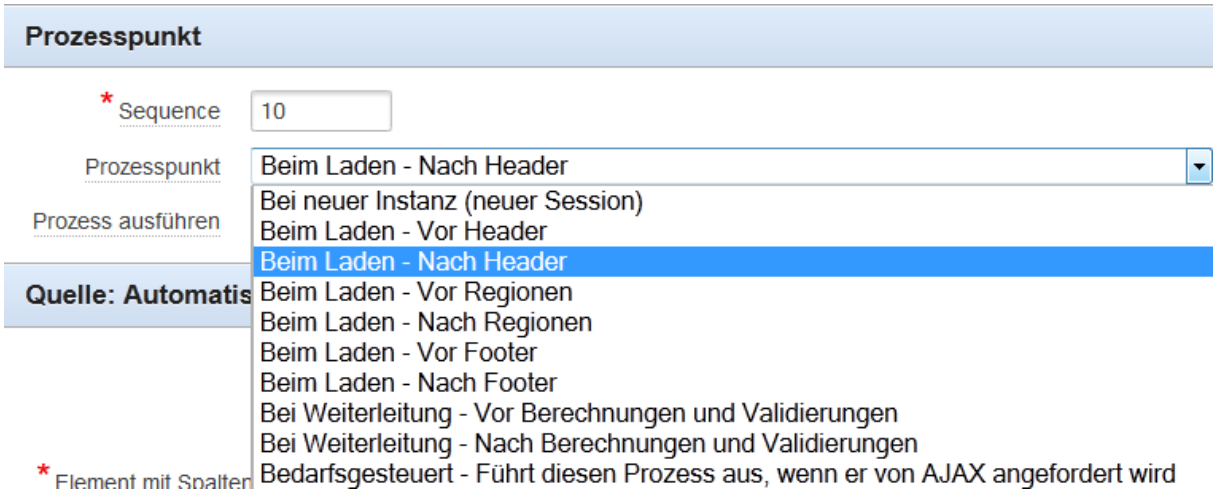
Neben der jeweiligen Seite die aktuell angezeigt wird, werden auch die Seite 0 / Global Page, sowie die evtl. vorhandenen Application Prozesse verarbeitet. Der Entwickler muss sich vor dem Hinzufügen einer Komponente also genau überlegen ob diese auf einer oder einzelnen Seiten benötigt wird oder mehrfach.
Selbst wenn eine mehrfache Nutzung vorhanden ist, lässt sich über eine saubere Kapselung von Logik in den Seiten oder in der Datenbank sehr viel optimieren.

Als Entwickler sollte man eine Entscheidung zwischen Performance und Wartbarkeit nicht nur auf Vermutungen oder in Zukunft evtl. kommenden Änderungen abstützen. Es ist durchaus sinnvoll eine mehrfach benutzte Logik einmalig abzulegen, aber nicht um jeden Preis auch nur einmalig auf zentral bei jedem Seitenaufruf zu benutzen.

## Die Technik

Beim Aufbau der Seiten arbeitet APEX alle Komponenten hinsichtlich einer vordefinierten Reihenfolge ab. Befindet man sich im Application Builder so wird bei einer Seite der Bereich der „Seitenwiedergabe“ (Page Rendering) von oben nach unten abgearbeitet. Dabei hilft es enorm, dass man die aktuelle Tree View benutzt, da nur dort die Komponenten hinsichtlich ihrer Reihenfolge für

den Entwickler dargestellt werden.
APEX  ermöglicht dabei dem Entwickler Prozesse, Berechnungen und Verzweigungen „Vor Header“, „Nach Header“, „Vor Regionen“, „Nach Regionen“, „Vor Footer“ und „Nach Footer“ auszuführen.




Dabei sollte der Entwickler versuchen eine Verarbeitung so gezielt wie möglich auszuführen, dass möglichst wenige Schritte in der richtigen Reihenfolge ausgeführt werden. Man sollte zum Beispiel nicht erst Daten laden, wenn andere Eigenschaften danach ausgewertet werden die dafür sorgen, dass auf eine andere Seite umgeleitet wird.
Auch sollte kein Laden von Daten stattfinden, wenn diese nicht angezeigt werden.

Entscheidender Punkt neben der Reihenfolge und der Abhängigkeit zwischen Prozessen ist der richtige Einsatz der Bedingungen / Conditions. Hier kann der Entwickler einerseits enorm Zeit sparen, wenn ein Prozess nur angestoßen wird wenn dieser wirklich benötigt wird. Neben der Performance ist eine gut definierte Bedingung auch eine Hilfe zur Identifikation der Abhängigkeiten zwischen Prozessen.

Man sollte sich aber im Klaren sein, dass Elemente die durch Bedingungen ausgeblendet werden, nicht nur nicht sichtbar sind, sondern komplett nicht im HTML vorhanden sind. Das führt natürlich dazu, dass die Seite anders aussieht, aber auch dazu, dass evtl. vorhandene Dynamic Actions diese Elemente nicht benutzen können (also auch nicht wieder einblenden). Zusätzlich sollte man dabei beachten, dass Application Items generell nicht mittels JavaScript verarbeitet werden können, da diese nicht im HTML vorhanden sind. Für Eigenschaften die seitenübergreifend vorhanden sein müssen und im JavaScript verarbeitet werden, müssen Elemente auf Seite 0 erstellt werden.

**Die Seitenverarbeitung**

Hier gelten bzgl. der Reihenfolge und Bedingungen natürlich die gleichen Vorgaben wie schon beim Seitenaufbau. Zusätzlich sollte man aber beachten, dass vor allem beim Einsatz von Verzweigungen (Branches) darauf geachtet wird, diese frühest möglich auszuführen. Teils sollte man eine Seitenverarbeitung auch gar nicht erst aufrufen, wenn eine Schaltfläche (Button) zum Beispiel auf eine andere Seite verzweigt, sollte dies direkt im Button hinterlegt werden.

**Nutzung von SQL, PL/SQL & Items**

Die große Auswahl an Möglichkeiten im Bereich der Bedingungen und Validierungen, sowie die unterschiedlichen Möglichkeiten auf den Wert eines Elements zuzugreifen birgt auch einige Besonderheiten. Am effektivsten ist die Auswertung eines Elements indem man die Einstellung „Wert

des Elements/Spalte …“ beim Vergleichen/Validieren verwendet. Im Vergleich zur Abfrage eines Element Wertes über SQL, PL/SQL oder weitere Mechaniken kann APEX am schnellsten intern selbst den Wert übersetzen. Benötigt man komplexere Validierungen sollte man darauf achten, dass man keinen Kontextwechsel von SQL und PL/SQL verwendet.
Innerhalb von APEX sollte man PL/SQL im Bereich der Validierungen und Bedingungen SQL vorziehen. Dabei sollte man auch darauf achten, dass man ein Element mittels „:P5_ITEM“ referenziert und wenn möglich auf den Zugriff mittels v-Syntax „v(‚P5_ITEM‘) verzichtet. Neben der Performance ist dieser Zugriff auch unsicherer.

**Statistiken & Debugging**

Will man allgemein wissen wo in der Anwendung es am schlimmsten aussieht, ist es sinnvoll und hilfreich die APEX internen Statistiken auszuwerten. Hierbei sollte man die Daten der Produktivumgebung auswerten die man im jeweiligen Workspace findet.
Loggt man sich ein befinden sich diese unter Administration -> Aktivität überwachen -> Analyse der Seitenansicht -> Seitenansicht pro gewichteter Seiten-Performance. Hierbei sollte man bei jedem Report darauf achten, dass man auf die jeweilige Anwendung filtert und den Zeitraum möglichst groß wählt um aussagekräftige Werte zu bekommen. Wurde in letzter Zeit ein Update eingespielt, welches die Performance anpasste sollte man exemplarisch den Zeitraum entsprechend anpassen, so dass die aktuell gültigen Werte beachtet werden oder man einen guten Vergleich zwischen den Versionen bekommt.

Mit diesen Zahlen bekommt man einerseits ein Bild welche Seite insgesamt am meisten benutzt werden und Informationen über die Performance der Seiten.

Falls auf der Produktion ein Zugriff auf die Anwendungsseiten besteht und ein Debugging aktiv ist, kann man am besten dort die Seite mit aktiviertem Debugging ausführen. Ist dies nicht der Fall sollte man versuchen die annähernd gleiche Datenkonstellation und Menge auf die Entwicklung zu bringen.

Bevor man nun die Seite ausführt kann man über die APEX Entwicklerleiste das „Debuggen“ aktiveren. Über „Debug anzeigen“ bekommt man nun die Übersicht der Aufrufe die mit aktiviertem „Debuggen“ durchgeführt wurden.
Beim Debuggen gibt es 2 unterschiedliche Aufrufe. Wird eine Seite aufgerufen so beginnt die Auflistung der einzelnen Prozesse mit dem ersten Prozess „SHOW“, wird eine Seite zum Server übertragen so beginnt es mit „ACCEPT“. In beiden Varianeten bekommt man alle Prozesse die APEX ausführt als Liste mit der verstrichenen Zeit in Millisekunden dargestellt und als kurze Übersicht wird eine Grafik mit einem Balkendiagramm benutzt in welcher man visuell schnell einen Überblick bekommt welcher Prozess wie viel Zeit benötigt. Durch die Prozesse bekommt man mit wie schnell APEX intern welche Prozesse ausführt. Dabei sollte man beachten, dass die Application Prozesse, die Prozesse der Seite 0 und die aktuelle Seite ausgeführt werden.
Will man eigene Einträge der Liste hinzufügen sollte man sich das Package APEX_DEBUG genauer anschauen.

**PL/SQL & SQL tuning**

Da komplexe Mechanismen nicht allein durch APEX bewerkstelligt werden und innerhalb APEX die richtige Anrfage maßgeblich zur Performance beiträgt ist der korrekte Umgang mit SQL und PL/SQL enorm wichtig. Zur besseren Auswertung der Queries sollten die Anfragen nicht in APEX selber hinterlegt werden, sondern über Views abgebildet werden, die man mittels des Ausführungsplans in der Datenbank wesentlich effektiver analysieren kann. Begriffe wie result_cache und determinstic functions sind richtig angewendet ein weiterer Baustein zur Performance.

Werden manche Daten komplex aufgearbeitet und dann in Interactive Reports verwende, ist die Nutzung von Collections oder Materialized Views zu empfehlen.

**JavaScript & Statische Dateien**

Neben der Datenbank sind die Elemente welche über den Webserver geliefert werden können bei Performance Fragen auch am besten auf dem Betriebssystem abzulegen. Da in APEX sehr viel dynamisch erstellt wird, ist das Caching nur richtig effektiv möglich wenn JavaScript und Grafiken auf dem Betriebssystem abgelegt werden. Caching, Kompression und Minimierung des JavaScripts können wertvolle Sekunden und auch Traffic sparen.

Zur Analyse von der Performance bei JavaScript kann der Firebug mit integriertem Debugger sowie der Darstellung von Ladezeiten der einzelnen Webseitenbestandteilen zusätzlich Transparenz schaffen wo noch getuned werden kann.

Zuletzt kann die richtige Anwendung von JavaScript mit PL/SQL als AJAX, welches asynchron benutzt wird zusätzlich dazu beitragen, dass einzelne Prozesse so aufgerufen werden, dass die Seite selber nicht blockiert wird.

**Fazit**

Wenn man weiß wie man Probleme finden kann und danach die Stelle des Problems genau lokalisiert kann man dem Endnutzer viel Ärger ersparen. Oft wird fällt eine einzige langsame Stelle in der Anwendung negativ auf die ganze Anwendung und APEX zurück und APEX wird als langsam bezeichnet. Um dies zu vermeiden lohnt sich eine effektive und schnelle Analyse der Anwendung so gut wie immer.

**Kontaktadresse:**

Oliver Lemm

MT-AG

Balcke-Dürr-Allee, 9

D-40882 Ratingen

| | |
|---|---|
| Telefon: | +49 (0) 2102-309 61 164 |
| Fax: | +49 (0) 2102-309 61 101 |
| E-Mail | oliver.lemm@mt-ag.com |
| Internet: | http://oliverlemm.blogspot.de |
| Twitter | https://twitter.com/OliverLemm |