„Jedes Byte zählt“ – Neue Tuning-Rezepte für APEX-Anwendungen in der Cloud

Andreas Wismann
WHEN OTHERS
D-41564 Kaarst

Schlüsselworte

APEX, Tuning, PL/SQL, SQL

Einleitung

APEX-Anwendungen sind so schnell, wie man es ihnen ermöglicht. Der Vortrag zeigt praxiserprobte Verfahren, um (insbesondere mobil genutzte) Web-Anwendungen unter Application Express einen gewissen „Schub“ zu verleihen. Sie erfahren, wie Sie einzelne APEX-Komponenten (Themes, Templates, Items, Reports, Shared Components, Skripte usw.) so optimieren können, dass am Ende ein Minimum an Bytes übertragen wird, ohne dabei den Anwendungskomfort zu opfern. Wenn Sie einmal die Tuning-Strategien kennen und bereit sind, einige "Tweaks" vorzunehmen, erhalten Sie schnelle und schlanke HTML5-Apps, die Ihr Browser auch unterwegs im Handumdrehen laden und ausführen kann.

Ist APEX nicht schon längst „Cloud“?

Wie bereits in der Konferenzbroschüre thematisiert wurde, ist „die Cloud“ im weitesten Sinne ein Teil des Internets, in dem Plattformen, Softwareprodukte, verteilte Dienste und Daten bereitgestellt werden. Zum Arbeiten ist man nicht wie früher „isoliert“, sondern im Netz und oft auch mobil unterwegs. APEX ist für diese Anwendungsfälle ein hervorragendes Entwicklungswerkzeugt, denn es nutzt (ausschließlich) Internet-Technologien. Außer wenn der Entwickler APEX auf seinem eigenen Rechner installiert hat, ist der Benutzer einer APEX-Anwendung typischerweise immer „online“ – sei es nun im Firmennetz oder eben im weltumspannenden Internet.

Ziel: Schlanker und schneller

Daraus folgt: Tuning-Verfahren, um Web-Anwendungen generell zu beschleunigen, lassen sich auch auf APEX-Anwendungen übertragen. Dabei sind aus Sicht des Entwicklers die folgenden Gesichtspunkte besonders interessant, da er sie unmittelbar beeinflussen kann:

1. Weniger Traffic generieren („Jedes Byte zählt“)
2. Daten schneller bereitstellen (Wartezeiten verringern)
3. Die „gefühlte“ Geschwindigkeit der Anwendung erhöhen

Freundlicherweise stellt APEX dem Entwickler eine fertige Infrastruktur bereit, mit der die wichtigsten Anwendungsfälle rund um „datenzentrierte Anwendungen“ hervorragend bedient werden können, ohne besondere Kenntnisse von Web-Technologien zu besitzen. Doch um welchen Preis?

Traffic: Themes und Templates

Nimmt man ein APEX-Seitentemplate genauer unter die Lupe, dann erkennt das geübte Auge auf den ersten Blick ein großes Optimierungspotenzial. Es befinden sich zuhauf Anweisungen im HTML-Gerüst, die „für alle Fälle“ und alle Browser ein zufriedenstellendes Ergebnis liefern. Der Preis dafür besteht darin, eine Menge zusätzlichen Code zu senden, angefangen bei

```
<!--[if lt IE 7 ]><html class="ie6 no-css3" lang="&BROWSER_LANGUAGE.">
<![endif]-->
<!--[if IE 7 ]><html class="ie7 no-css3" lang="&BROWSER_LANGUAGE.">
<![endif]-->
<!--[if IE 8 ]><html class="ie8 no-css3" lang="&BROWSER_LANGUAGE.">
<![endif]-->
<!--[if IE 9 ]><html class="ie9" lang="&BROWSER_LANGUAGE."> <![endif]-->
<!--[if (gt IE 9)|!(IE)]><!--><html lang="&BROWSER_LANGUAGE.">
<!--<![endif]-->
...
<!--[if lte IE 6]><div id="outdated-browser">
#OUTDATED_BROWSER#</div><![endif]-->
```

Man ahnt auch als nicht-HTML-Experte, dass es hier um die Behandlung der unterschiedlichen Programmversionen des Internet Explorers geht. Auch ein großer Teil des Codes in den CSS-Dateien, die anschließend an den Browser gesendet werden, bewerkstelligt eben diese Fallunterscheidungen.

Welche Alternativen existieren?

Sofern es irgendwie möglich ist, vereinbaren Sie mit Ihren Anwendern „den“ Referenz-Browser. Ob es sich dabei um ein Produkt von Microsoft oder einen der zahlreichen Konkurrenten handelt sei dahingestellt. Hauptsache es wird eine Auswahl getroffen. Anschließend können Sie den nicht mehr benötigten Overhead an „Was-wäre-wenn“-Code aus dem Template entfernen (siehe Demonstration im Vortrag). Die Browserfestlegung vereinfacht also nicht nur das Testen der Anwendung, sondern schont auch die Übertragungsbandbreite erheblich, wenn man sie konsequent umsetzt.

Label Templates

Immer wenn Ihre Anwendungsseiten aus vielen Page Items bestehen, lohnt sich auch das Abspecken der Label Templates. Beispiele dafür werden im Vortrag gezeigt. Hier Bytes zu sparen, wirkt sich 1:1 auf das zu übertragende Datenvolumen aus, denn der Browser kann in Webformularen prinzipbedingt kein HTML cachen.

Anstatt des originalen Label-Templates

```
<label for="#CURRENT_ITEM_NAME#" tabindex="999"><a class="optional-w-help"
href="javascript:popupFieldHelp('#CURRENT_ITEM_ID#','&SESSION.')"
tabindex="999"><a></label>
```

wird dieses verwendet:

```
<label for="#CURRENT_ITEM_NAME#" class="h"></label>
```

Bei 20 Page Items sparen Sie so pro Seitenaufruf(!) allein durch diese Reduktion des Label Templates bereits etwa 2 Kilobyte an Traffic ein. Sie müssen lediglich einmalig eine Dynamic Action oder - besser - eine JavaScript-Funktion in einem Seitenskript hinzufügen, die den Aufruf der Hilfefunktion übernimmt. Wie das geht, erfahren Sie im nächsten Abschnitt.

jQuery anstatt Dynamic Actions?

Dynamic Actions sind ein Segen, denn seit APEX 4.0 sind auch nicht-JavaScript-Programmierer in der Lage, clientseitiges Verhalten sogar weitgehend deklarativ zu programmieren. Selbst der Autor, in der Rolle als erfahrener jQuery-Programmierer, ist manchmal schneller mit der Aufgabe fertig indem er eine DA verwendet. Die Kosten für dieses Feature sind zusätzliche Bits und Bytes, und zwar nicht unerheblich viele. Der clevere Mechanismus, mit dem die Dynamic Actions funktionieren, erfordert

eine Codebasis, die bei jedem Seitenaufruf zum Browser gesendet wird. Entwickler, die jQuery beherrschen, können hier erheblich zur Code-Ersparnis beitragen: Das Skript zum Laden des Hilfefensters, welches zum obigen Seitentemplate gehört, lautet:

```
$(function(){
$('label.h').click(function(){
  popupFieldHelp($(this).next('input').attr('id'),$v('pInstance'))})
});
```

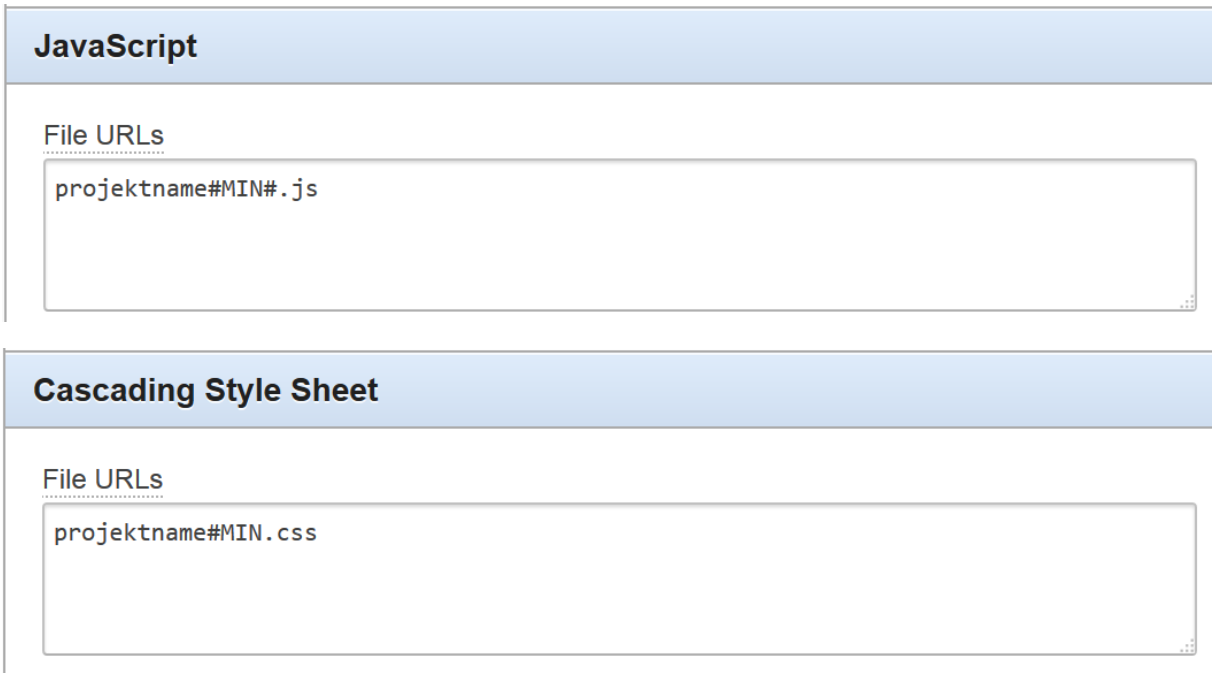
Bereits ab dem zweiten Page Item verbraucht das weniger Quellcode als zwei ungekürzte Label Templates. Es lässt sich theoretisch sogar noch etwas kürzer notieren, worunter allerdings die Lesbarkeit leidet.

Code Caching

APEX bietet an vielen Stellen Eingabefenster an, in denen sich clientseitiger Code (HTML, JavaScript, CSS) unterbringen lässt. Verwenden Sie eigene jQuery-Skripte oder CSS-Deklarationen? Dann sind diese Eingabefenster ideal, um den Code zu schreiben, zu testen und zu ändern. Doch sobald Ihre Zeilen erst einmal produktionsreif sind, sollten Sie sie in zwei Dateien überführen:

- eine Datei „projektname.js“
- eine Datei „projektname.css“

Diese beiden Dateien werden im Serververzeichnis Ihrer APEX-Installation platziert (typsicherweise in einem Ordner namens /i/) und aus dem Seitentemplate heraus referenziert:



Datei-Minifizierung

Sowohl JavaScript- als auch Stylesheet-Dateien lassen sich auf Syntaxbasis komprimieren. Lassen Sie den folgenden Textinhalt einer APEX-JavaScript-Datei rein visuell auf sich wirken:

```
[]}function bk(a,b){var c;if(b.nodeType===1){b.clearAttributes&&b.clearAttri
b.mergeAttributes(a),c=b.nodeName.toLowerCase();if(c==="object")b.outerHTML=
a.type!=="radio"){if(c==="option")b.selected=a.defaultSelected;else if(c==='
a.checked&&(b.defaultChecked=b.checked=a.checked),b.value!==a.value&&(b.valu
{if(b.nodeType===1&&!!f.hasData(a)){var c,d,e,g=f._data(a),h=f._data(b,g),i=
i)for(d=0,e=i[c].length;d<e;d++)f.event.add(b,c+(i[c][d].namespace?".":"")+i
(h.data=f.extend({},h.data))}}function bi(a,b){return f.nodeName(a,"table")?
[0]||a.appendChild(a.ownerDocument.createElement("tbody")):a}function U(a){v
```

Bei diesem Code handelt es sich um minifiziertes JavaScript, welches von sämtlichen syntaktisch überflüssigen Elementen wie Leerzeichen, Kommentaren, langen Variablennamen und Parameterbezeichnungen befreit ist. Anders als bei physikalischer Komprimierung (z.B. dem unspezifischen „Zippen“ einer Datei) wird hier also mit entsprechendem Knowhow an der Quelle komprimiert: Der Algorithmus kennt die Regeln und Besonderheiten der verwendeten Skriptsprache. Ein so komprimiertes Skript „macht dasselbe“ wie der zuvor von Hand geschriebene Code, kommt aber mit bis zu drei Viertel weniger Code aus. Eine solche Minifizierung sollten Sie auch Ihren eigenen Skripten gönnen, bevor sie produktiv gehen.

Unkomplizierte Komprimier-Tools, die Ihren Code online bearbeiten, finden Sie beispielsweise hier:

- http://jscompress.com
- http://csscompressor.com

Ein großer Vorteil der Minifizierung von JavaScript-Dateien ist außerdem, dass Sie dem neugierigen Benutzer Ihrer Anwendung im HTML-Quellcode der APEX-Seite nicht sofort im Klartext mitteilen, was Sie da eigentlich hinzuprogrammiert haben… Sie werden zwar keinen „JavaScript-Geheimcode“ ausführen, doch grundsätzlich gibt man Geschäfts- oder Eingabelogik ja nur ungern preis.

Preloader

Es gehört zum Handwerkszeug eines erfahrenen Webseitenprogrammierers, dass er die benötigten Assets (hauptsächlich Skriptdateien und Grafiken) möglichst im Voraus lädt, damit sie sofort einsatzbereit sind, sobald eine Seite sie zum ersten Mal benötigt. In APEX-Anwendungen, die meist eine Anmeldekennung erfordern, drängt sich hierfür die Login-Seite geradezu auf: Der Benutzer ist sowieso mit dem Eintippen seiner Kenndaten beschäftigt und wird das Laden von Dateien im Hintergrund nicht bemerken oder als Verzögerung empfinden. Im APEX-Standardtemplate für die Login-Page wird das bereits ausgenutzt – referenzieren also auch Sie Ihre eigenen Anwendungsdateien und -bilder dort, letztere in einem unsichtbaren Seitenelement. Der Browser wird diese Dateien weisungsgemäß laden und cachen und kann so auf den folgenden Seiten direkt auf den Cache zurückgreifen, anstatt erst „bei Bedarf“ zu laden.

Achten Sie aber auf eine gute Balance zwischen Vorausladen und tatsächlicher Verwendung. Nicht jede einzelne Grafik einer riesigen Webanwendung gehört vorausgeladen, sondern nur diejenigen, die mit hoher Wahrscheinlichkeit bei jeder Session abgerufen werden, sei sie noch so kurz. Das sind beispielsweise Ihre Standard-Icons oder die Firmenlogos in unterschiedlichen Größen.

Mit all diesen Optimierungen verhält es sich ähnlich wie mit den Anstrengungen in der Formel 1: Nicht die einzelne Maßnahme bringt den Erfolg, sondern die Summe aller Vorkehrungen. Wie im Vortrag gezeigt wird, können Sie bei cleverem Vorgehen eine ganz erhebliche Menge an Datenvolumen und Übertragungszeit einsparen. Da die meisten mobilen Datentarife immer noch nach Volumen abrechnen und die Übertragungsraten je nach Gebiet weiterhin stark variieren, können Sie bei Ihren Anwendern mit schlanken und schnellen APEX-Anwendungen ganz erheblich punkten.

„Gefühlte Geschwindigkeit“

Oft lässt es sich gar nicht vermeiden, dass APEX ein lang laufendes SQL aufrufen muss, beispielsweise wenn jeden Morgen ein umfangreicher Report über die Vorgänge im Data Warehouse abgerufen werden soll. Was passiert, wenn der Benutzer eine APEX-Seite mit einem Report-Langläufer aufruft, ist uns allen bekannt: Entweder die Seite baut sich zunächst überhaupt nicht auf, oder Teile werden bereits dargestellt und man wartet auf die Daten. Usability-Studien (und die Erfahrung aus diversen Projekten) haben ergeben, dass die Benutzer solche Wartezeiten nicht nur als unangenehm empfinden, sondern sie auch dem verwendeten Anwendungs-Tool zuschreiben – in diesem Fall eben APEX.

Ein eleganter Trick, um dieses Dilemma zu umgehen, besteht darin, dem Benutzer einen echten Fortschrittsbalken anzubieten. „Echt“ bedeutet in diesem Fall: Der Benutzer bekommt nicht nur eine sich bewegende Grafik angezeigt (dies machen APEX Interactive Reports ja bereits von selbst). Stattdessen wird ein wachsender Fortschrittsbalken dargestellt, der im Idealfall auch eine Prozentanzeige mitliefert und - bei 100% angekommen - auch wirklich „fertig“ ist. Im Anschluss daran wird der neue Report angezeigt. Durch diesen Kunstgriff erreichen Sie, dass bei objektiv gleicher Wartezeit der Benutzer einerseits viel weniger Ungeduld entwickelt (anscheinend macht das Betrachten eines Fortschrittsbalkens mehr Spaß als auf einen sich drehende blaue Sanduhr zu schauen), und obendrein ordnet er die Wartezeit offenbar „instinktiv“ der angesprochenen Datenbank zu, nicht mehr APEX. Und so stimmt es ja immerhin auch ☺

Einen solchen Fortschrittsbalken können Sie entweder mit Hilfe des Package DBMS_SCHEDULER programmieren, oder Sie verwenden die Methoden des Package APEX_PLSQL_JOB, welches ein Wrapper um das Standard-Package DBMS_JOB ist.

Die Präsentation

Wie ein solcher Fortschrittsbalken implementiert wird, außerdem viele andere Live-Beispiele zum Thema „Tuning von APEX-Anwendungen“, sehen Sie in meinem Vortrag auf der DOAG 2014 am Dienstag, 18.11., um 11:00 Uhr im „Raum St. Petersburg“.

Kontaktadresse:

Andreas Wismann
WHEN OTHERS
Hirschstraße 10
D-41564 Kaarst

Telefon: +49 (0) 2131-314 9966
E-Mail: wismann@when-others.com
Internet: when-others.com
Twitter: @whenothers