Eine der Stärken von Apex – die einfache Anpassung des Layouts – ermöglicht schon heute das einfache Erstellen mobiler Anwendungen für Smartphones und Tablets. Dieser Artikel zeigt die Anforderungen mobiler Anwendungen und deren technische Umsetzung.

Apex für mobile Clients

Peter Raganitsch, click-click IT Solutions

Ein starker Trend der letzten Jahre – nicht zuletzt wegen des Erfolgs der iPhones und iPads – ist die immer stärker zunehmende Verbreitung von mobilen Endgeräten. Dies zieht sich vom „kleinsten" Mitarbeiter bis hinauf in die Vorstandsetagen durch – nahezu jeder hat sein Smartphone oder Tablet. Auch Oracle ist sich der Tatsache bewusst geworden, dass in der heutigen Zeit mobile Clients immer mehr zunehmen, und wird – laut Statement of Direction (siehe Seite 14) – mit Apex Version 4.2 nativen Support für mobile Anwendungen liefern. Aufgrund der fast grenzenlosen Flexibilität von Apex ist es schon jetzt mit Version 3 und Version 4 möglich, Anwendungen zu erzeugen, die sich auch für mobile Endgeräte eignen beziehungsweise für diese optimiert sind.

Erwartungen an mobile Anwendungen

Dieser Abschnitt enthält nicht die umzusetzenden Punkte einer bestimmten Anwendung, sondern wir überlegen, was generell von einer mobilen Anwendung geleistet werden soll. Punkt eins ist die für die Bildschirmgröße optimierte Darstellung für die Bildschirmgröße: Der wohl größte Unterschied zwischen normalen Desktop-Anwendungen und mobilen Anwendungen ist die Größe des Bildschirms. Wird auf einem Smartphone-Browser eine normale Website geladen, dann sieht das aus wie in Abbildung 1. Eine optimierte Darstellung für mobile Endgeräte zeigt auf den ersten Blick die wesentlichen Informationen in der entsprechenden Größe, ohne dass dafür gezoomt werden muss (siehe Abbildung 2).

Hier ist also einerseits darauf zu achten, die Darstellung der Inhalte in der richtigen Größe auszugeben. Andererseits ist genau zu überlegen, welche Information wirklich übermittelt werden soll. Weniger ist an dieser Stelle sicher mehr. Durch das Weglassen unnötiger Information kann der Blick auf das Wesentliche gelenkt werden.

Ein weiterer Punkt ist die einfache Eingabe. Da man auf einem Smartphone nur über eine virtuelle Tastatur die Eingabe durchführen kann, ist es umso wichtiger, dem Benutzer die Eingabe so einfach als möglich zu machen. Mit der Definition der zu erwartenden Eingabe in einem Feld über den HTML-Input-Typ und einen entsprechenden Platzhalter gibt man dem Browser die Möglichkeit die Eingabe zu vereinfachen. Dies geschieht üblicherweise über eine

Abbildung 1: Darstellung einer normalen Webseite auf einem Smartphone-Browser

Abbildung 2: Darstellung einer optimierten Webseite

Abbildung 3: Eine Virtuelle Tastatur für Text

```
<!DOCTYPE html>
<html lang="&BROWSER_LANGUAGE." xmlns="http://www.w3.org/1999/xhtml" xmlns:htmldb="http://htmldb.oracle.
com" xmlns:apex="http://apex.oracle.com">
<head>
  <title>#TITLE#</title>
  <link rel="icon" href="#IMAGE_PREFIX#favicon.ico" type="image/x-icon">
  <link rel="shortcut icon" href="#IMAGE_PREFIX#favicon.ico" type="image/x-icon">
  #HEAD#
  <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">
  <link rel="stylesheet" href="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.css" />
  <script src="http://code.jquery.com/jquery-1.6.4.min.js"></script>
  <script src="http://code.jquery.com/mobile/1.0/jquery.mobile-1.0.min.js"></script>
</head>
<body #ONLOAD#>
  <div id="Page&APP_PAGE_ID." data-role="page">
    #FORM_OPEN#
```

Listing 1

```
      <div data-role="header" data-backbtn="false">
        <h1>#TITLE#</h1>
        #NAVIGATION_BAR#
      </div><!-- /header -->
      <div id="messages">#SUCCESS_MESSAGE##NOTIFICATION_
MESSAGE##GLOBAL_NOTIFICATION#</div>
      <div data-role="content">
        #REGION_POSITION_01#
        #REGION_POSITION_02#
        #REGION_POSITION_03#
        #REGION_POSITION_04#
        #REGION_POSITION_05#
        #REGION_POSITION_06#
        #REGION_POSITION_07#
        #BOX_BODY#
      </div><!-- /content -->
```

Listing 2

```
      <div data-role="footer">
        <h4>#REGION_POSITION_08#</h4>
      </div><!-- /footer -->

    #FORM_CLOSE#
  </div>
</body>
</html>
```

Listing 3

spezielle virtuelle Tastatur für den jeweiligen Datentyp. So gibt es Beispielsweise eigene Tastaturlayouts für Text, Zahlen, Telefonnummern, E-Mail-Adressen oder URLs (siehe Abbildung 3).

Ein besondere Augenmerk gilt der einfachen Auswahl. Aufgrund der begrenzten Bildschirmgröße ist es notwendig, Auswahlfelder, Links und Schaltflächen so groß wie möglich zu gestalten, um deren Auswahl zu vereinfachen. Man denke nur an eine turbulente Zugfahrt, während der man versucht, auf einem Smartphone-Browser einen bestimmten Link auszuwählen. Mit einem großen Finger einen kleinen Link auf einem ebenso kleinen Bildschirm zu treffen, ist nicht einfach.

Apex wird mobil

Der Markt für Smartphones und Tablets ist riesig, das trifft nicht nur auf die Menge der Käufer zu, sondern auch auf die Anzahl der Hersteller, Modelle und Betriebssysteme. Jedes Gerät hat gewisse Vor- und Nachteile, vor allem aber unterschiedliche Features. Obwohl die meisten davon HTML5 und CSS3 schon umsetzen können, gibt es doch gewisse Feinheiten, die überall anders sind. Das beginnt bei Bildschirmgröße und Auflösung und endet mit der Interpretation von HTML5/CSS3-Standards.

Dass man sich da nicht selbst um die Feinheiten der Darstellung kümmern will, dürfte verständlich sein. Stattdessen bedienen wir uns zu diesem Zweck eines optimierten Frameworks für mobile Webseiten. Entschieden haben wir uns für „jQuery mobile“ (http://jquerymobile.com/), da dies perfekt zu dem bereits in Apex integrierten Basis-Framework jQuery passt, unter den verfügbaren mobilen Frameworks am reifsten ist und das breiteste Spektrum an Endgeräten abdeckt. Ganz nebenbei sei bemerkt, dass auch Oracle zu diesem Schluss gekommen ist und jQuery mobile als Basis für den mobilen nativen Support in Apex 4.2 verwenden wird.

Erstellen mobiler Templates in Apex

Der erste Schritt, um unsere Apex-Anwendung mobil zu machen, ist die Er-

stellung der entsprechenden Templates. Was wir genau in die Templates schreiben müssen, – wie die HTML-Struktur auszusehen hat – ist in der Dokumentation von jQuery mobile unter http://jquerymobile.com/demos/1.0.1/docs/pages/page-anatomy.html beschrieben. Wir werden daraus jetzt einen kleinen Teil umsetzen. Damit hat der Entwickler allerdings alle Informationen zur Verfügung, um weitere Templates nach der Vorgabe in der Dokumentation zu erzeugen. Dazu erstellen wir zunächst ein neues Page-Template in den Shared Components, nennen dieses „Mobile Page“ und fügen diese Code-Zeilen ein: Im Bereich „Header“ siehe Listing 1, im „Body“ siehe Listing 2 und im „Footer“ des Page-Template siehe Listing 3.

Ein Page-Template in Apex kann so kurz und übersichtlich aussehen. Damit haben wir den generellen Seitenaufbau definiert und jQuery mobile auf unserer Seite eingebunden. Für eine vollwertige Applikation fehlen noch Regions-, Listen- und Report-Templates, die wir sogleich anlegen. Listing 4 zeigt das Region-Template „Mobile Region“, Listing 5 das Listen-Template „Mobile Liste“ Before Rows und Listing 6 das „Current/Non-Current“-List-Element. Für „After Rows“ wird zum Schluss „
“ eingefügt. Beim Anlegen des Report-Template „Formatted Text“ ist es wichtig, den Typ „Named Column (row template)“ zu wählen (siehe Listing 7).

```
<div id="#REGION_STATIC_ID#" #REGION_ATTRIBUTES#>
  <h2>#TITLE#</h2>
  <div data-role="controlgroup" data-type="horizontal">#CLOSE##PREV
IOUS##NEXT##DELETE##EDIT##CHANGE##CREATE##CREATE2##EXPAND##COPY##HE
LP#</div>
  <div>
    #BODY#
  </div>
</div>
```

Listing 4

```
<ul data-role="listview" data-inset="true">
```

Listing 5

```
<li  #IMAGE_ATTR#><a href="#LINK#" rel="external"
#A01#>#TEXT#</a></li>
```

Listing 6

```
<li>
  <a href="#LINK#" rel="external">
    <h3>#TITLE#</h3>
    <p><strong>#BOLD_TEXT#</strong></p>
    <p>#PLAIN_TEXT#</p>
  </a>
</li>
```

Listing 7

Der Grund, aus dem wir für Reports dieses spezielle Template erstellen, hängt mit den beengten Platzverhältnissen auf Smartphones zusammen. Die von Desktop-Browsern gewohnte Report-Darstellung in tabellarischer Form scheidet aufgrund des Platzmangels aus. Stattdessen verwenden wir eine datensatzorientierte Darstellung, die im jQuery-mobile-Jargon „Listview“ heißt. Dabei wird nicht ein generisches Template für Spalten definiert, sondern ein Template für den gesamten Datensatz. Das gibt uns wiederum volle Kontrolle über das Layout. Einzige Bedingung für „Named Column“-Report-Templates ist, dass das Report-Select-Statement genau jene Spalten liefert, die laut den Template-Platzhaltern definiert wurden. Wer jetzt verwirrt ist, wird weiter unten beim Report-Beispiel hoffentlich Klarheit bekommen.

Mit der Erstellung dieser Basis-Templates haben wir die Möglichkeit, unsere mobile Apex-Anwendung mit genau den gleichen Mitteln zu erstellen, wie wir dies auch für eine normale Desktop-Apex-Anwendung tun würden.

Erstellen der ersten mobilen Apex-Anwendung

Zug um Zug entsteht nun die erste mobile Apex-Anwendung, die sich aus einer Startseite mit Hauptmenü, einer Report-Seite und einer Formular-Seite zusammensetzt. Für die Startseite (Page 1) mit dem Hauptmenü erstellen wir in den Shared Components eine Apex-Liste, die zumindest einen einzigen Eintrag „Customer“ enthält, der auf Page 10 verweist. Selbstverständlich verwenden wir für die erstellten Elemente (Seite, Region, Liste) die zuvor erstellten Templates. Abbildung 4 zeigt das Ergebnis der soeben erstellten Seite.

Weiter geht es mit Seite 10. Diese wird erstellt mit einem klassischen Apex-Report (nicht als interaktiver, sondern als normaler Bericht) unter Verwendung des zuvor erstellten Named-Column-Template. Das Select-Statement des Berichts basiert auf dem Datenmodell der Apex Sample Application und liefert genau jene Spalten, die im Report-Template erwartet werden (siehe Listing 8).

Will man an dieser Stelle noch mehr oder andere Spalten in den Bericht aufnehmen, muss das Template entsprechend erweitert werden. Die Namen der Platzhalter kann man dabei frei ver-

```
SELECT ,f?p=&APP_ID.:11:&APP_SESSION.::::P11_CUSTOMER_ID:'||CUSTOMER_ID AS LINK
     , cust_last_name || ,, , || cust_first_name                        AS TITLE
     , CUST_STREET_ADDRESS1 || decode(CUST_STREET_ADDRESS2, null, null, ,, , || CUST_STREET_ADDRESS2) AS
BOLD_TEXT
     , cust_city ||', ,|| cust_state ||', ,|| cust_postal_code          AS PLAIN_TEXT
  FROM DEMO_CUSTOMERS
 ORDER BY CUST_LAST_NAME
        , CUST_FIRST_NAME
```

Listing 8

geben, die einzige Bedingung ist das Übereinstimmen von Spaltennamen und Platzhalter (siehe Abbildung 5).

Die erste Spalte des Berichts liefert eine URL zurück. Diese wird verwendet, um vom Übersichts-Bericht auf das Detail-Formular zur Bearbeitung des Datensatzes zu verzweigen. Entsprechend der Link-Definition im Bericht erstellen wir nun eine neue Seite 11 mit einem Formular auf Tabelle „DEMO_CUSTOMERS“. Da wir die nötigen Definitionen zum Aussehen der Seite bereits in den Templates erledigt haben, sehen wir sofort das Ergebnis (siehe Abbildung 6).

Beim Durchgehen der einzelnen Eingabefelder sehen wir, dass in allen Feldern dieselbe virtuelle Tastatur eingeblendet wird. Wollen wir spezielle Tastaturen (wie oben beschrieben) verwenden, müssen wir uns vorerst eines Plug-ins bedienen. Unter http://www.apex-plugin.com/oracle-apex-plugins/item-plugin/html5-input-item_107.html kann das freie Plug-in „HTML5 Input Item“ heruntergeladen werden. Dies unterscheidet sich vom regulären Text-Feld durch zwei zusätzliche Einstellungen. Einerseits kann der HTML5-Typ des Eingabefelds (Text, E-Mail, Telefon, URL etc.) gesetzt werden, andererseits auch ein Platzhalter. Mit dem HTML5-Typ wird die virtuelle Tastatur gesteuert, der Platzhalter zeigt einen Werte-Vorschlag an, falls das Eingabefeld noch leer ist. Diese beiden Einstellungen erleichtern dem Benutzer die Eingabe und zeigen, in welcher Form man den Eingabewert erwartet.

Navigation Bar

Bisher wurde noch nicht erklärt, wie die Buttons in die Kopfzeile kommen. Dort steht links neben dem Seitentitel ein Button „Back“ und rechts neben dem Titel ein Button „Logout“. Beides sind typische Elemente einer Apex Navigation Bar und diese wollen wir dafür auch verwenden. Dazu fügen wir in unserem Page-Template im Bereich „Navigation Bar“ folgenden Code ein:

```
„<a href="#LINK#"
rel="external" #ALT#>#TEXT#</
a>“.
```

Bei der Definition der Navigation Bar Entries können wir nun im Feld „Icon Image Alt“ Zusatz-Attribute für unsere Buttons definieren. Beim Button „Back“ steht dort: „data-icon="arrow-l"“. Der Button „Logout“ wird ergänzt durch „data-icon="delete" class="ui-btn-right"“.

Durch Setzen dieser Zusatzattribute im HTML-Code steuern wir jQuery mobile. Mit „data-icon“ geben wir an, welches Icon bei dem Button angezeigt werden soll (eine vollständige Liste finden sie in der Dokumentation von jQuery mobile), mit dem Setzen der CSS-Klasse „ui-btn-right“ stellen wir sicher, dass der Button rechtsbündig angezeigt wird. Auch dies ist in der Dokumentation entsprechend beschrieben.

Im Grunde können wir mit dieser Vorgangsweise alle Features von jQuery mobile entsprechend in unserer Apex-Applikation abbilden. Einfach die in der Dokumentation beschriebene HTML-Struktur im Template abbilden, schon haben wir das gewünschte Ergebnis.

Abbildung 4: Die Startseite mit dem Hauptmenü

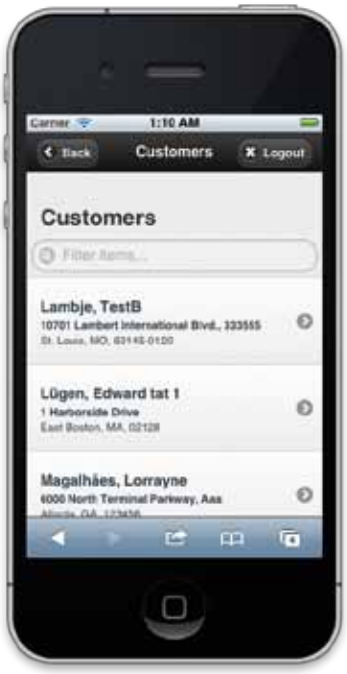

Abbildung 5: Apex-Bericht in mobiler Ansicht

Abbildung 6: Apex-Formular in mobiler Ansicht

Abbildung 7: Das Hauptmenü mit den Icons

Layout-Möglichkeiten

Durch den Einsatz von jQuery mobile haben wir natürlich weiterhin die Möglichkeit, das Aussehen unserer Applikation zu beeinflussen. So kann durch Definition eines sogenannten „Swatch" eines der mitgelieferten Farb-Schemata von jQuery mobile aktiviert werden. Dazu einfach das „data-theme" im HTML-Code hinzufügen und auf den Wert a,b oder c setzen. Damit können die 3 mitgelieferten Farbschemata aktiviert werden, Details dazu in der Dokumentation unter http://jquerymobile.com/demos/1.0.1/docs/pages/pages-themes.html. Aber auch abseits von jQuery mobile ist eine Gestaltung des Layouts möglich. Allein durch Verwendung eigener Listen-Templates können sehr ansehnliche Darstellungen erzeugt werden (siehe Abbildung 7). Dazu verwenden wir eine normale Apex-Liste mit Icon-Definition und ein eigenes Listen-Template. Eine genauere Beschreibung davon sprengt an dieser Stelle allerdings den Rahmen des Artikels.

Aussicht auf Oracle Apex 4.2

Oracle selbst darf natürlich noch nicht verraten, was genau in dem kommenden Release enthalten sein wird und wann dieses erscheint (siehe Seite 14), deswegen an dieser Stelle ein paar Mutmaßungen des Autors. Oracle Apex 4.2 enthält native Unterstützung für die Erstellung mobiler Applikationen. Dies wird über jQuery mobile realisiert und in einer Mischung aus Plug-ins (für Items und für Listendarstellungen) und Templates zur Verfügung gestellt. Das Ziel an dieser Stelle wird sicherlich die einfache Anwendung mit größtmöglicher Flexibilität sein. Eine Demo-Anwendung und ein eigenes mobiles Theme werden an dieser Stelle ebenso erwartet. Im Bereich „Charts für mobile Anwendungen" wird vermutlich die neueste Version von AnyChart integriert, die anstelle von Flash mit HTML5 arbeitet und somit auf allen Plattformen funktioniert.

Fazit

Die kommende Version von Oracle Apex wird einen großen Schwerpunkt auf das Thema „Mobile" legen. Allerdings müssen wir nicht darauf warten und können schon jetzt mobile Anwendungen erstellen. Da wir dazu dieselben Mittel verwenden, wie Apex das in Zukunft auch machen wird, ist diese Investition auch zukunftssicher.

Peter Raganitsch
click-click IT Solutions
peter.raganitsch@click-click.at

Wir begrüßen unsere neuen Mitglieder

Persönliche Mitglieder

Annegret Schlenker
Albin Hollenstein
Sascha Schmorde
Jerome Witt
Gregory Steulet
Marek Adar
David Hüber
Harald Unterschütz
Ralph Mösch
Hans Frötsch
Denis Heinzmann
Andreas Risch
Matthias Laudt
Fredi Schweizer
Marion Erlebach
Wolfram Ditzer
Detlef Pollkläsner
Michael Podewils
Andreas Wismann
John Bintz
Sergey Semenov
Ralf-Peter Schaum
Claus-Dieter Seidel
Piotr Giemza
Olaf Czekay
Ulrich Küsters
Volker Eckert
Kurt Stadlmair

Firmenmitglieder

Jörg Biesewig, megatel I.-u. K.systeme
Martin Wunderli, Trivadis
Thilo Rottach, TeamBank
Jan Gorkow, SD&C
Thomas Eifert, RWTH Aachen