Building Mobile Web Applications with Oracle Application Express (APEX)

Marc Sewtz
Senior Software Development Manager
Oracle Corporation
New York

Key Words

Oracle Application Express, Mobile Web Development, HTML5, jQuery Mobile

Introduction

Oracle Application Express (APEX) is the native Web application development framework for the Oracle Database. Since its inception in 2004 as a feature of Oracle Database 10g, Oracle Application Express has seen tremendous uptake, with hundreds of thousands of downloads, and thousands of customers with secure, scalable web applications deployed into production. Oracle Application Express is a standard component of Oracle Database.

With the tremendous growth mobile Internet usage has seen over the past few years and the widespread adoption of mobile devices, building applications that work both on mobile devices as well as desktop web browsers, becomes increasingly important. Oracle Application Express 4.2 accommodates the need of customers to roll out their APEX applications to users who today are using an ever-larger variety of devices to access their systems by introducing mobile development capabilities; HTML5 based charting options and responsive themes and templates.

Starting with desktop computers to small laptops, ultra books, tablets and a plethora of smartphones, building an application that works well and looks good on all these devices is a challenge. And depending on the type of application there are different solutions to consider. Developers can choose to build native applications for a specific target platform, such as iOS devices, Android or Windows Mobile. This has the advantage of being able to fully exploit the features available on the device and getting the best possible performance. Alternatively developers can choose to build Web applications, which run in standard mobile web browsers available on mobile devices and have the advantage that they are built using standard web technology such as HTML, CSS and JavaScript. There are also hybrid applications, which combine elements of both native and web based solutions.

The focus of this paper is on building mobile Web applications on the Oracle database, using Oracle's Application Express (APEX) framework along with jQuery Mobile. Mobile Web applications aren't all that different from standard Web applications or any regular Web page for that matter. In fact, most mobile devices today allow opening most any Web page with their built-in mobile browsers, and depending on the device and OS used, those pages can be more or less usable, even without any mobile-specific treatment. Thus any application built for the Web, including those built with APEX, can be used on mobile devices. Using a mobile development framework, such as jQuery Mobile however allows for building Web pages, or Web applications that are designed specifically for the smaller screen size of mobile devices and that can take advantage of the device specific capabilities, which are typically not found in desktop browsers. This paper provides an overview of mobile Web application development, the jQuery Mobile framework, and how jQuery Mobile has been integrated into Oracle Application Express to allow for easy, wizard-driven development of mobile business applications on the Oracle database.

Mobile Web Applications

Mobile applications are built to run on mobile devices, like smart phones, personal digital assistants or tablet devices. They are optimized for a smaller screen and take advantage of the unique capabilities of mobile devices, providing a touch-enabled interface, and responding to gestures and change in screen orientation. There are two types of mobile applications, native applications, which are compiled programs that run natively on the device and mobile Web applications, which run in a Web browser on the device. Native applications are the kind of applications that can be downloaded from an app store like Apple's iTunes, or the Android Marketplace. They have some advantages over mobile Web applications; they are fast and have access to all the capabilities of the device they are built for, like GPS, camera, etc. However they also have an important limitation: they are not portable, i.e. native mobile applications need to be built for a specific platform, such as iOS, Android, Blackberry, etc.

Mobile Web applications on the other hand are built using standard, cross-platform HTML, CSS and JavaScript, and thus run on any modern mobile device that provides a Web browser. So as a developer there's only one code base to maintain and when using a platform abstraction layer, such as Phone-Gap, it is even possible to make these applications appear as native applications, i.e. they can be included on the application launch pad of the device, they can gain access to some of the capabilities typically reserved for native apps and they can perform faster than pure mobile Web applications due to the option of storing some application components, like JavaScript libraries, images and CSS files on the device. They do however require an Internet connection to be available at all times, as most of the page rendering and application logic happens on the server.

The difference between a standard Web site and a mobile Web application is primarily the screen size for which it is developed. Most Web applications that are developed for desktop browsers are optimized to work well on screens that use a minimum resolution of 1024x768 pixels. Using these types of applications on a Smartphone is difficult because it requires the user to zoom in on certain portions of the Web site.

Figure 1: Screen size and aspect ration on different devices

This is where mobile Web applications come in; mobile Web applications are developed to work well on much smaller screen sizes. Much of the content typically found on pages designed for desktop browsers is omitted on pages designed for mobile browsers; this includes large company logos, extensive navigational controls, sidebars, large advertisement banners, etc. In addition to the amount of content that can be rendered on a smaller screen, there are additional differences to consider, mobile devices generally can be used in vertical or horizontal orientation, and most have touch-enabled screens, which requires page buttons, form fields, etc. to be large enough to be accessible by touching them with a finger. Many devices also support gestures like swiping to navigate forward and backward and scrolling through long lists of data.

Building mobile Web applications allows customers to use these applications on mobile devices. And although developers have only one codebase to maintain, there are significant differences in how mobile Web browsers render pages. As with desktop browsers, different mobile browsers interpret HTML and CSS differently. Devices like the iPhone, iPads and Androids use a Web browser based on the Webkit rendering engine. Microsoft's Windows Mobile platform uses a rendering engine based on the desktop version of Internet Explorer browser. Blackberry adopted Webkit for version 6 and above of their operating system, but devices using older versions of the Blackberry OS lack support for certain HTML5 and CSS3 features, and the list goes on. So as difficult as it is to develop cross-browser compatible Web applications for the desktop, developing cross-browser compatible applications for mobile applications gets even harder.

Considering the mobile specific requirements for touch interfaces, screen orientation and navigation as well as the difficulties building applications for different mobile browsers, developers are well advised to consider using a mobile development framework that already provides built-in support for the capabilities of mobile devices and that is optimized to support a wide variety of mobile browsers. While iOS and Android are certainly the most popular mobile platforms today, other platforms like Blackberry are still widely used in the business community. So coding mobile Web applications for only the most popular platforms might lock out large groups of potential users and might ultimately lead a customer to reject a mobile application that has only limited device support.

jQuery Mobile

One of the mobile development frameworks available today is jQuery Mobile. jQuery Mobile aims to provide a unified user interface that works seamlessly across all popular mobile device platforms. jQuery Mobile is based on jQuery and jQuery UI. Web pages in jQuery Mobile are built on a foundation of clean, semantic HTML to ensure compatibility with pretty much any Web-enabled device. In devices that interpret modern CSS and JavaScript, jQuery Mobile applies progressive enhancement techniques to unobtrusively transform the semantic page into a rich, interactive experience that leverages the power of jQuery and CSS.

Key feature of jQuery Mobile

- Built on jQuery core for familiar and consistent jQuery syntax and minimal learning curve.
- Compatible with all major mobile & desktop platforms: iOS, Android, Blackberry, Windows Mobile, Opera Mobile/Mini, Firefox Mobile and all modern desktop browsers.
- Lightweight size (~20k for all mobile functionality) and minimal file dependencies for speed.
- HTML5 Markup-driven configuration of pages and behavior for fast development and minimal required scripting.
- Progressive enhancement approach brings core content and functionality to all mobile, tablet and desktop platforms and a rich, native app-like experience on newer mobile platforms.
- Automatic initialization by using HTML5 data-role attributes in the HTML markup to act as the trigger to automatically initialize all jQuery Mobile widgets found on a page.
- Accessibility features such as WAI-ARIA are included to ensure that the pages work for screen readers (e.g. VoiceOver in iOS) and other assistive technologies.
- Touch and mouse event support streamlines the process of supporting touch, mouse, and cursor focus-based user input methods with a simple API.
- UI widgets enhance native controls with touch-optimized, theme-able controls.
- Powerful theming framework and Theme Roller application make highly-branded experiences easy to build.

Mobile development with Oracle Application Express

Oracle Application Express has long been known for its declarative, wizard-driven approach to software development and its fast and lightweight user interface. So when Oracle looked to build mobile development capabilities into APEX 4.2, the goal was to find a mobile framework that would complement established concepts and integrate well with existing frameworks. jQuery Mobile was an ideal choice. Sharing the same base libraries (jQuery and jQuery UI) that were already included with APEX in past releases added only a minimal amount of additional code to download into the browser, and being almost entirely HTML5 driven made it very easy to build new themes and templates without having to add a large amounts of new custom JavaScript code.

Figure 2: APEX User Interfaces

User Interfaces

APEX renders pages by merging HTML templates with content from the Oracle database. Each page uses several different types of templates. Starting with the page template, region templates, and templates for reports, labels, buttons, etc. The templates that are associated with an application are organized as a user interface theme and APEX allows for switching from one theme to another which essentially replaces all templates of the current theme with the templates of the new theme. In past releases APEX would only allow one active theme for an application. With APEX 4.2 this has been extended to allow for multiple themes, with each theme associated with one user interface. A user interface describes the target platform for a page. APEX 4.2 ships with built-in support for desktop user interfaces and mobile user interfaces. Thus each application can have up to two user interfaces, one for desktop and one for mobile. Each page of an application is associated with one user interface, so the templates available to be used on that page are limited to the templates of the corresponding user interface theme.

User interfaces allow for having applications that have both mobile pages and desktop pages. This makes it possible to share common components, like authentication schemes, application items processes, access to session state, etc. The user interface attributes define which user interface is to be used as the default user interface when the application is first loaded into the browser. Alternatively it is also possible to enable auto-detection, so that depending on the device used, the user is taken to the appropriate login and home pages. The user interface of an application can be selected by the user in the create application wizard. Existing applications that have only one user interface can be extended to include additional user interfaces. New user interfaces can be added on the application attributes page or the create theme wizard. When a new user interface is added, the developer first selects the user interface type and then selects a theme from a list of available themes for the user interface. Adding a user interface also creates a corresponding login page and a global page for content that is shared across all pages associated with the same user interface.

Global Page

Content that is defined on a global page, such as regions, items and buttons, is included on all pages that share a user interface with the global page. In previous versions of Application Express this global page was called page 0. With the support for multiple user interfaces, a single page 0 was no longer practical, so it is now possible to define one global page for each user interface. This global page can have any arbitrary page number, i.e. the page number does not necessarily have to be 0.

Mobile Theme

Desktop and mobile themes are similar in that the various templates all use standard HTML along with APEX specific substitution strings to define the look and feel of pages, region, buttons, etc. The mobile theme simply loads an additional JavaScript library, the jQuery Mobile library when the page is rendered. This library is looking for certain HTML5 data- attributes in the page's HTML code. In the minimal page sample shown below, there are data-role="page", data-role="header" and date-role="content" defined. These attribute along with the DIV structure provide sufficient information for jQuery Mobile to render this page as a mobile page, using the default jQuery Mobile look & feel.

```
<!DOCTYPE html>
<html>
  <head>
    <title>My Page</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link rel="stylesheet" href="/mobile/jquery.mobile-1.1.0.min.css" />
    <script src="/jquery-1.7.1.min.js"></script>
    <script src="/mobile/jquery.mobile-1.1.0.min.js"></script>
  </head>
  <body>
    <div data-role="page">
      <div data-role="header">
        <h1>My Title</h1>
      </div><!-- /header -->
      <div data-role="content">
        <p>Hello world</p>
      </div><!-- /content -->
    </div><!-- /page -->
  </body>
</html>
```

Every mobile page template follows a similar structure, using additional data- attributes for roles, colors positioning, etc. The look and feel of a jQuery Mobile page is defined in the jQuery Mobile CSS file. This file can be overwritten with custom styles and colors to create a branded user experience. Tools such as ThemeRoller make the creation of a customized look and feel very easy.

List Views

jQuery Mobile pages typically use HTML lists to display most content. Components like menus, reports, drill-down list, etc. are generally rendered as a single or multi level HTML list. JQuery Mobile adds styling to these lists, defining whether they list items stretch the full with of the browser window or are framed as an inset list. It is possible to optionally enable search, define drill down links, position content and include images and icons. For mobile devices, list views are a good alternative to drill down reports, like the form and report on a table typically used in desktop applications. The create application and create page wizard expose the list view and form option when creating mobile applications or adding content to existing mobile pages.

Transitions

The jQuery Mobile framework includes a set of CSS-based transition effects that can be applied to any page link or form submission with Ajax navigation. As the user navigates from page to page, submits forms, clicks on links and buttons, the content of the next page is loaded through AJAX into the DOM tree of the current page. Once loaded a transition effect is applied to replace the content of the current page with the content of the next page. Transition effects include slide, fade, pop, flip and others. Transitions can be disabled, resulting in a standard full page load navigation. Disabling transitions can be configured for the whole page or individual links. Especially when including links to external sites, or desktop pages within the current application, non-AJAX based navigation should be used.

New Item Types

APEX 4.2 adds new item types specifically for mobile and updates some existing items types with mobile specific enhancements. As with other jQuery Mobile and HTML5 components, there's a fall back rendering built-in for devices that can't display certain controls.

- Select lists have been extended to show lists of values using the device's native LOV control for LOV with up to 10 values. LOVs with more than 10 values are rendered as popup dialogs
- A new HTML5 date picker allows users to edit date values with the device's native calendar control.
- A slider control allows for setting a value based on moving a slider across a range of values
- The Yes / No control is an alternative to checkboxes that provides a touch friendly way to turn attributes on and off
- Text fields include a number of subtypes that prompt the mobile device to display a keyboard layout specific to the subtype. The email subtype includes "@" and "." alongside the regular non-numeric characters on the keyboard, the URL subtype shows a ".com" instead and the phone subtype display a numeric phone keypad.

HTML5 Charts

APEX 4.2 introduces HTML5 charts. This cross-browser and cross-platform charting solution is intended for everybody who deals with creation of dashboard, reporting, analytics, statistical, financial or any other data visualization solutions. Based on an upgraded Anychart 6.0 charting engine, it is now possible to have several of the charts types that were available in earlier releases of APEX to be rendered in HTML5 (SVG) rather than Adobe Flash, thus allowing those charts to be displayed on mobile devices that's do not support Flash or no longer support Flash. When creating new charts, developers have a choice to create the chart as Flash-preferred or HTML5-only. Flash-preferred charts are rendered as Flash charts when Flash is available and otherwise fall back to HTML5. HTML5-only charts are always rendered in HTML5 format. Existing charts can be upgraded to support Flash-preferred and HTML5.

Testing and Deployment

jQuery Mobile works on desktop as well as mobile Web browsers. In order to test a mobile APEX application, or an APEX application with mobile pages, most functionality can be verified by simply using a modern Web browser on the desktop such as Safari or Chrome. Safari and Chrome provide the closest approximation to the mobile user experience, as both are based on the Webkit layout engine – just like the most commonly used Mobile browsers on iOS and Android.

Using a desktop browser has some limitations though, as some functionality is only available on Mobile devices and Mobile device simulators like those available with Apple's XCode development framework or the Android SDK. This includes some of the new jQuery Mobile and HTML5 item types discussed earlier, response to orientation change and swipe events, and having Flash-preferred charts fall back to HTML5 mode. So any testing strategy should also include mobile device simulators as well as testing on actual Mobile devices.

Once a Mobile APEX application is ready for roll out to the end users, deployment works similar to desktop applications. If development and production use is done on different systems, the application first gets installed on the production system and then the URL of the application is shared with the end users. This URL can be bookmarked and when running on Mobile devices, also be added to the device's home screen. Adding a bookmark to the home screen makes the APEX application appear alongside other applications that are installed on the device and allows for quick and easy access to the APEX application in the future.

About the Author

Marc Sewtz is a Senior Software Development Manager at Oracle Corporation in New York. With over sixteen years of industry experience, Marc held roles in Consulting, Sales and Product Development and now manages a global team of Software Developers and Product Managers in the Oracle Application Express (APEX) development team, part of the Oracle Database Tools group. Marc and his team are responsible for features such as the development of Mobile Web Applications with APEX, Reporting and Charting, Tabular Forms, Oracle Forms to APEX conversion and integration with Oracle Business Intelligence Publisher. Marc has a Master's degree in computer science from the University of Applied Sciences in Wedel, Germany.

Contact Information:
Marc Sewtz
Oracle Corporation
120 Park Avenue, 26th Floor
New York, NY 10017
USA

Phone:	+1 212 508 7951
E-Mail	marc.sewtz@oracle.com
Blog:	http://marcsewtz.blogspot.com
Internet:	apex.oracle.com