1

Introducing the iPhone and iPod touch Development Platform

The introduction of the iPhone and subsequent unveiling of the iPod touch revolutionized the way people interacted with handheld devices. No longer did users have to use a keypad for screen navigation or browse the Web through "dumbed down" pages. These Apple devices brought touch screen input, a revolutionary interface design, and a fully functional Web browser right into the palms of people's hands. However, the question in the developer community in the months leading up to the release of the iPhone was: Will Apple allow third-party developers to develop custom applications for this new mobile platform? Apple's response was one that made Web developers happy and Objective-C programmers sad — iPhone and iPod touch applications would be Safari-based apps that are built using standard Web technologies. Apple opted for this solution as a way to provide developers with the freedom to create custom apps, all the while maintaining control of the user experience of these two devices.

Discovering the Mobile Safari Platform

An iPhone and iPod touch application runs inside of the built-in Safari browser that is based on Web standards, including:

- HTML/XHTML (HTML 4.01 and XHTML 1.9, XHTML mobile profile document types)
- CSS (CSS 2.1 and partial CSS3)
- JavaScript (ECMAScript 3, JavaScript 1.4)
- AJAX (e.g., XMLHTTPRequest)
- Ancillary technologies (video and audio media, PDF, and so on)

Safari on iPhone and iPod touch (which I refer to throughout the book as *Mobile Safari*) becomes the platform upon which you develop applications and becomes the shell in which your apps must operate (see Figure 1-1).

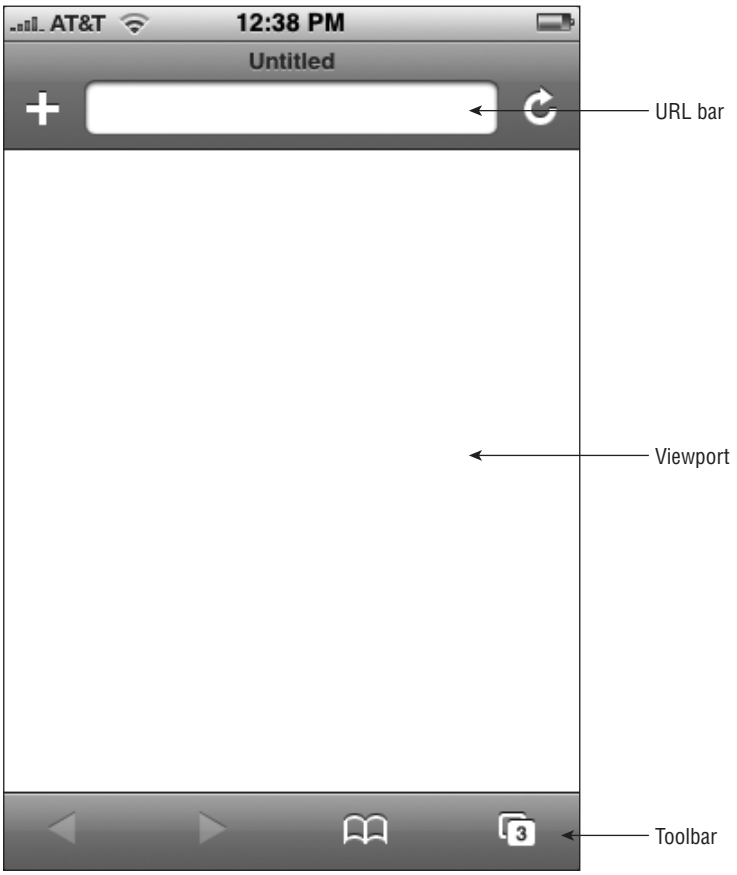

Figure 1-1: Mobile Safari user interface

Mobile Safari is built with the same open source WebKit browser engine as Safari for OS X and Safari for Windows. However, while the Safari family of browsers is built on a common framework, you'll find it helpful to think of Mobile Safari as a close sibling to its Mac and Windows counterparts, not an identical twin to either of them. Mobile Safari, for example, does not provide the full extent of CSS or JavaScript functionality that its desktop counterpart does.

In addition, Mobile Safari provides only a limited number of settings that users can configure. As Figure 1-2 shows, users can turn off and on support for JavaScript, plug-ins, and a pop-up blocker. Users can also choose whether they want to always accept cookies, accept cookies only from sites they visit, or never accept cookies. A user can also manually clear the history, cookies, and cache from this screen.

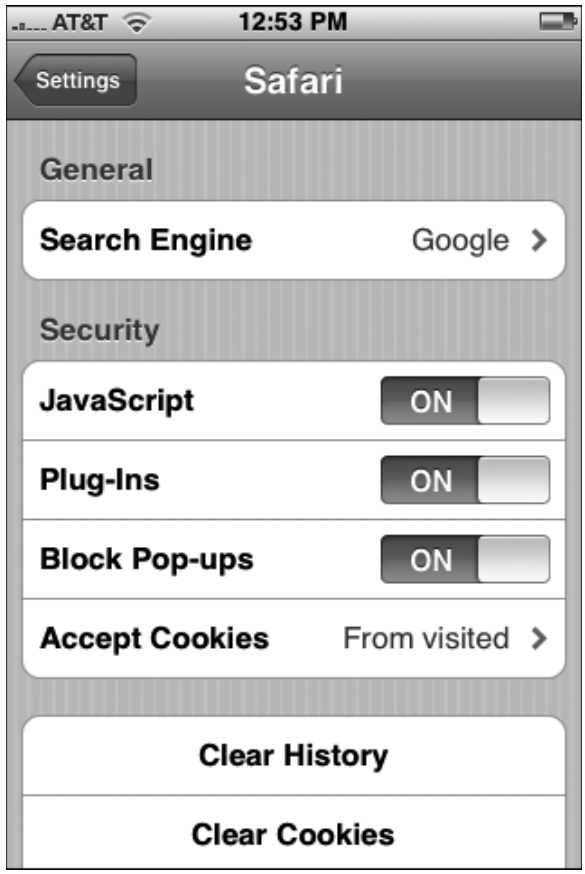

Figure 1-2: Mobile Safari preferences

Quite obviously, there are important differences between an iPhone/iPod touch application running inside of Mobile Safari and a native application. From a developer standpoint, the major difference is the programming language — utilizing Web technologies rather than Objective-C. However, there are also key end-user implications, including:

- *Performance:* The performance of a Safari-based application is not going to be as responsive as a native compiled application, both because of the interpretive nature of the programming languages as well as the fact that the application operates over Wi-Fi and EDGE networks. (Remember, iPod touch supports Wi-Fi access only.) However, in spite of the technological constraints, you can perform many optimizations to achieve acceptable performance. (Several of these techniques are covered in Chapter 10.)

Table 1-1 shows the bandwidth performance of Wi-Fi and EDGE networks.

Table 1-1: Network Performance

Network	Bandwidth
Wi-Fi	54 Mbps
EDGE	70–135 Kbps, 200 Kbps burst

- *Launching:* While the built-in applications are all launched from the main Springboard screen of the iPhone and iPod touch (see Figure 1-3), Web developers do not have access to this area for their applications. Instead, a user can only access your application by entering its URL or by selecting a bookmark from the Bookmarks list (see Figure 1-4). Unfortunately, there is absolutely nothing a Web developer can do to emulate the native application launch process.

Figure 1-3: Built-in applications launch from the main Springboard.

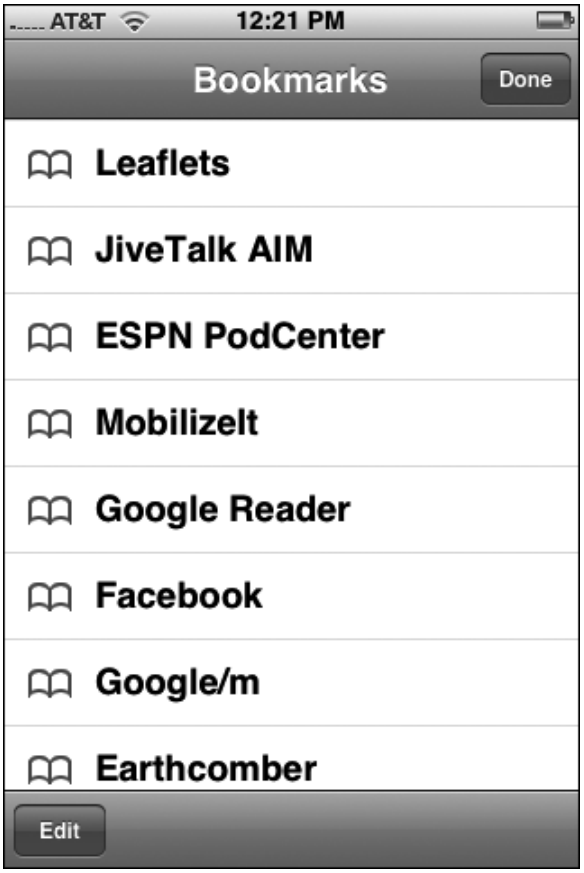

Figure 1-4: Web applications launch from the Bookmarks list.

- *User interface (UI):* The built-in iPhone and iPod touch applications adhere to very specific Apple UI design guidelines. As Chapters 3 and 4 explain in detail, you can closely emulate native application design using a combination of HTML, CSS, and JavaScript. The only constraint to complete emulation is the ever present bottom toolbar in Mobile Safari. Figures 1-5 and 1-6 compare the UI design of a native application and a Safari-based application.

Figure 1-5: Edge-to-edge navigation pane in the iPod app

Figure 1-6: Edge-to-edge navigation pane in a custom application

Four Ways to Develop for iPhone and iPod touch

A Web application that you can run in any browser and an iPhone/iPod touch application are certainly made using the same common ingredients — HTML, CSS, JavaScript, and AJAX — but they are not identical. In fact, there are four approaches to developing for iPhone and iPod touch:

- *Level 1: Fully compatible Web site/application:* The ground level approach is to develop a Web site/app that is "iPhone/iPod touch–friendly" and is fully compatible with the Apple mobile devices (see Figure 1-7). These sites avoid using technologies that the Apple mobile devices do not support, including Flash, Java, and other plug-ins. The basic structure of the presentation layer also maximizes use of blocks and columns to make it easy for users to navigate and zoom within the site. This basic approach does not do anything specific for iPhone/iPod touch users, but makes sure that there are no barriers to a satisfactory browsing experience. (See Chapter 8 for converting a Web site to be friendly for iPhone and iPod touch users.)

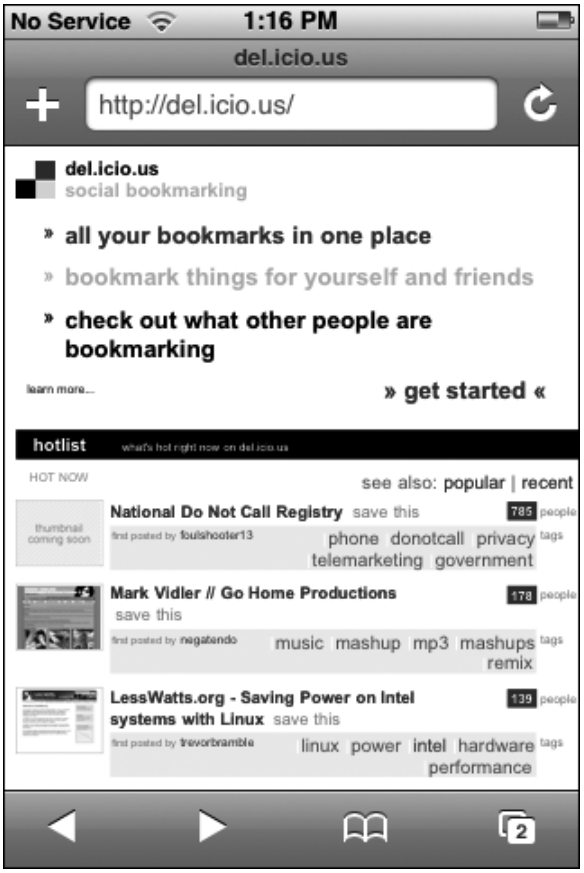

Figure 1-7: Site is easy to navigate.

- *Level 2: Web site/application optimized for Safari:* The second level of support for iPhone and iPod touch is to not only provide a basic level of experience for the Mobile Safari user, but also to provide an optimized user experience for users who use Safari browsers, such as utilizing some of the enhanced WebKit CSS properties supported by Safari.
- *Level 3: Dedicated iPhone/iPod touch Web site/application:* A third level of support is to provide a Web site tailored to the viewport dimensions of the iPhone and iPod touch and provide a strong

Web browsing experience for Apple device users (see Figures 1-8 and 1-9). However, while these sites are tailored for iPhone/iPod touch viewing, they do not always seek to emulate Apple UI design. And, in many cases, these are often stripped-down versions of a fuller Web site or application.

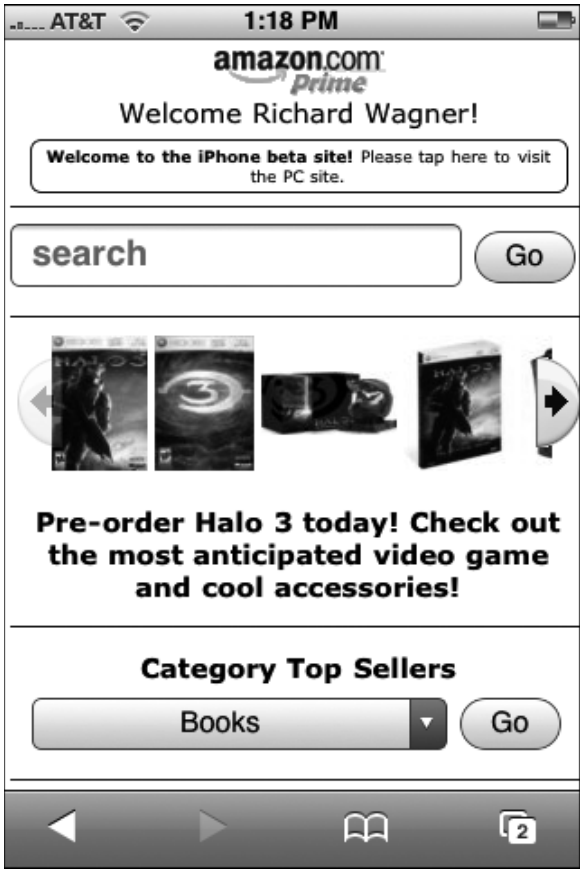

Figure 1-8: Amazon's iPhone site

Figure 1-9: Facebook closely emulates Apple UI design.

- *Level 4: Native-looking iPhone/iPod touch application:* The final approach is to provide a Web application that is designed exclusively for iPhone and iPod touch and closely emulates the UI design of native applications (see Figure 1-10). One of the design goals is to minimize user awareness that they are even inside of a browser environment. Moreover, a full-fledged iPhone application will, as is relevant, integrate with iPhone-specific services, including Phone, Mail, and Google Maps.

Therefore, as you consider your application specifications, be sure to identify which level of user experience you wish to provide iPhone and iPod touch users and design your application accordingly. In this book, I'll focus primarily on developing native-looking applications.

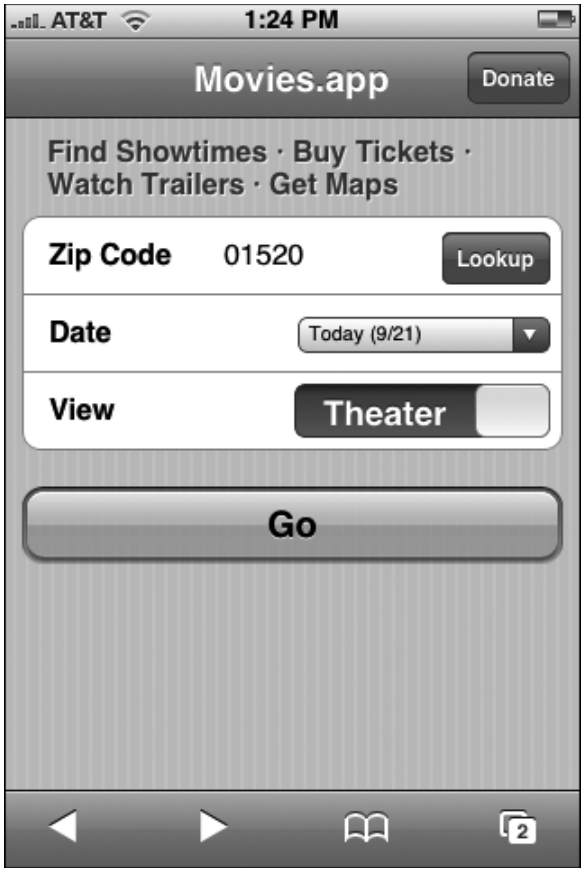

Figure 1-10: Application emulating Apple UI design

The Finger Is Not a Mouse

As you develop applications for iPhone and iPod touch, one key design consideration that you need to drill into your consciousness is that *the finger is not a mouse*. On the desktop, a user can use a variety of input devices — such as an Apple Mighty Mouse, a Logitech trackball, or a laptop touchpad. But, on screen, the mouse pointer for each of these pieces of hardware is always identical in shape, size, and behavior. However, on iPhone and iPod touch, the pointing device is always going to be unique. Ballerinas, for example, will probably input with tiny, thin fingers, while NFL players will use big, fat input devices. Most of the rest of us will fall somewhere in between. Additionally, fingers are also not nearly as precise as mouse pointers are, making interface sizing and positioning issues very important, whether you are creating an iPhone/iPod touch–friendly Web site or a full-fledged iPhone/iPod touch application.

Additionally, finger input does not always correspond to a mouse input. A mouse has a left click, right click, scroll, and mouse move. In contrast, a finger has a tap, flick, drag, and pinch. However, as an application developer, you will want to manage what types of gestures your application supports. Some

of the gestures that are used for browsing Web sites (such as the double-tap zoom) are actually not something you want to support inside of an iPhone and iPod touch application. Table 1-2 displays the gestures that are supported on iPhone and iPod touch as well as an indication as to whether this type of gesture should be supported on a Web site or application. (However, as Chapter 5 explains in detail, you will not have programmatic access to managing all of these inputs inside of Mobile Safari.)

Table 1-2: Finger Gestures

Gesture	Result	Web site	App
Tap	Equivalent to a mouse click	Yes	Yes
Drag	Moves around the viewport	Yes	Yes
Flick	Scrolls up and down a page or list	Yes	Yes
Double-tap	Zooms in and centers a block of content	Yes	No
Pinch open	Zooms in on content	Yes	No
Pinch close	Zooms out to display more of a page	Yes	No
Touch and hold	Displays an info bubble	Yes	No
Two-finger scroll	Scrolls up and down an `iframe` or element with CSS `overflow:auto` property	Yes	Yes

Finally, several mouse actions have no finger touch equivalents on iPhone and iPod touch. These include:

- No right-click
- No text selection
- No cut, copy, and paste
- No hover
- No drag-and-drop (though I offer a technique to roughly emulate it in Chapter 5)

Limitations and Constraints

Since iPhone and iPod touch are mobile devices, they are obviously going to have resource constraints that you need to be fully aware of as you develop applications. Table 1-3 lists the resource limitations and technical constraints. What's more, certain technologies (listed in Table 1-4) are unsupported, and you will need to steer away from them when you develop for iPhone and iPod touch.

Table 1-3: Resource Constraints

Resource	Limitation
Downloaded text resource (HTML, CSS, JavaScript files)	10MB
JPEG images	128MB (all JPEG images over 2MB are subsampled—decoding the image to 16x fewer pixels)
PNG, GIF, and TIFF images	8MB (in other words, `width*height*4<8MB`)
Animated GIFs	Less than 2MB ensures that frame rate is maintained (over 2MB, only first frame is displayed)
Non-streamed media files	10MB
PDF, Word, Excel documents	30MB and up (very slow)
JavaScript stack and object allocation	10MB
JavaScript execution limit	5 seconds for each top-level entry point (`catch` is called after 5 seconds in a `try/catch` block)
Open pages in Mobile Safari	8 pages

Table 1-4: Technologies not Supported by iPhone and iPod touch

Area	Technologies not supported
Web technologies	Flash media, Java applets, SOAP, XSLT, SVG, and Plug-in installation
Mobile technologies	WML
File access	Local file system access
Text interaction	Text selection, Cut/Copy/Paste
Embedded video	In-place video (tapping an embedded element will put iPhone/iPod touch into video playback mode)
Security	Diffie-Hellman protocol, DSA keys, self-signed certificates, and custom x.509 certificates
JavaScript events	Several mouse-related events (see Chapter 5)
JavaScript commands	`showModalDialog()`, `print()`
Bookmark icons	`.ico` files
HTML	`input type="file"`, tool tips
CSS	Hover styles, `position:fixed`

Accessing Files on a Local Wi-Fi Network

Since iPhone and iPod touch do not allow you to access the local file system, you cannot place your application directly onto the device itself. As a result, you need to access your Web application through another computer. On a live application, you will obviously want to place your application on a publicly accessible Web server. However, testing is another matter. If you have a Wi-Fi network at your office or home, I recommend running a Web server on your main desktop computer to use as your test server during deployment.

If you are running Mac OS X, you already have Apache Web server installed on your system. To enable iPhone and iPod touch access, go to System Preferences ⇨ Sharing ⇨ Services and turn the Personal Web Sharing option on (see Figure 1-11). When this feature is enabled, the URL for the Web site is shown at the bottom of the window. You'll use this base URL to access your Web files from iPhone or iPod touch.

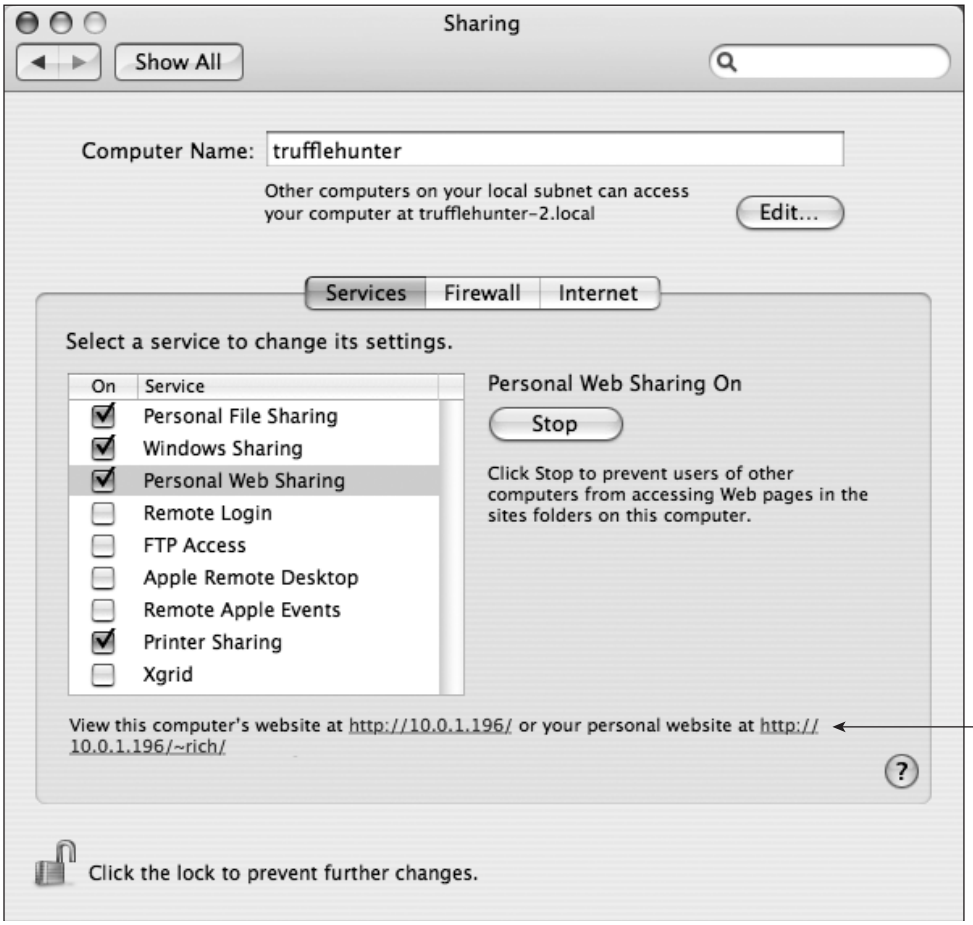

Figure 1-11: Turn on Personal Web Sharing.

You can add files either in the computer's Web site directory (`/Library/WebServer/Documents`) or your personal Web site directory (`/Users/`*`YourName`*`/Sites`) and then access them from the URL bar on your iPhone or iPod touch (see Figure 1-12).

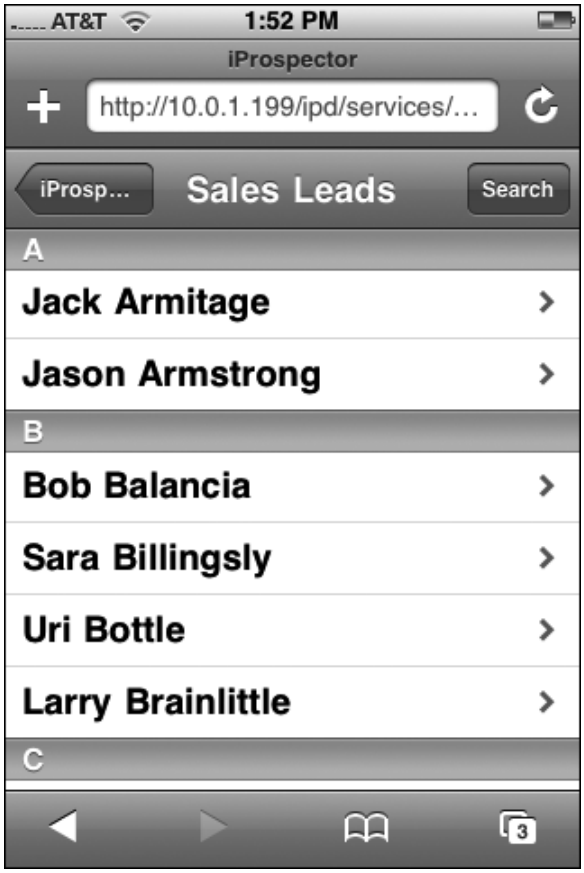

Figure 1-12: Accessing desktop files from iPhone

If your users experience crashing or instability inside Mobile Safari, direct them to clear the cache by tapping the Clear Cache button in the Safari Settings pane.