Chapter 6

Delivering Different Designs to Different Devices

In This Chapter
▶ Directing mobile devices
▶ Designing for particular mobile devices
▶ Planning a mobile site

In the spacious kitchen of Spago Beverly Hills — the flagship restaurant of Chef Wolfgang Puck — each gourmet dish is carefully prepared to satisfy the supermodels and billionaires that dine there. The chefs are happy to accommodate their every wish.

In contrast, just a few miles away at the crowded take-out counter at the Wolfgang Puck Café at the Los Angeles County Airport, what you see in the little plastic lunch packages is what you get. The staff at this busy airport café work quickly and efficiently, but the goal is to help travelers get their food in a hurry — no one here has the time or resources to handle special requests.

As you consider the range of options for designing a mobile Web site for all the different mobile devices that may visit your pages, it may be helpful to use this culinary example for comparison. At the high end of the mobile Web design spectrum, companies such as Microsoft have implemented mobile design strategies akin to the fine dining experience at Spago. Developers put tremendous resources into creating a complex system that delivers different versions of the Microsoft Cloud Computing site (shown in Figure 6-1), optimized for each device that visits the site. At the low end of the list of mobile design options, other companies create just one alternate version of their desktop Web site, stripped down and streamlined to meet the basic needs of mobile devices, much like the prepackaged meals sold at the Wolfgang Puck Café.

In this chapter, we review what it takes to deliver different versions of your Web site to different mobile devices. In the first section, we describe a few key concepts and discuss how to develop such a system. In the second part
of the chapter, we explore in detail how to plan and design a site that meets the needs of many different devices. To illustrate these concepts, we use the Microsoft Cloud Web site and include comments, interviews with designers, and screenshots of planning documents and designs. Figure 6-1 gives you an idea of what went into this complex mobile site.

![Figure 6-1: The Microsoft Cloud Computing site was designed to be displayed correctly in a variety of mobile devices.]

**Detecting and Directing Mobile Devices**

Restaurants must get the right meal to the right table while the food is still hot, or they won’t be in business long. To meet these goals, well-informed waitstaff must write the order correctly, and staff in a well-organized kitchen must prepare variations of the same meal quickly, without compromising quality.

Similarly, if you want to deliver tailored versions of your mobile Web site, you must properly detect what kind of device each visitor is using (device detection, which is like getting the order right), and then deliver the best version of your site for each device (content adaptation, which is like preparing each meal to order).

Before we delve into the details of device detection, let’s begin with an overview. If you want to deliver different versions of a Web site to different devices, follow these general steps:
1. Develop device profiles or parameters.

You first have to determine which devices you want to design your site to work well on. Since there are thousands of mobile phones and other devices available, it’s generally best to categorize devices into a set of profiles or develop a set of parameters (such as image sizes) that you can mix and match, based on things like the screen size and multimedia support of each device.

2. Develop (or acquire) a device database or a configuration file.

To deliver the right version of your site to the right device, you need a configuration file or database that contains detailed information about every device. Many private companies create their own device databases, but you can also use the open-source WURFL (Wireless Universal Resource File), described in the “Using open-source device detection scripts and services” section, later in this chapter.

3. Write (or acquire) a program that can deliver the right version of your site to the right device.

The challenge here is to develop an application that can match the contents of the device database with the different design options (the device profiles, special URLs, or other parameters) that you’ve created for your site. Top mobile design firms create their own solutions, but you find information about open-source solutions, including Andy Moore’s device detection solution (shown in Figure 6-2) in the “Using open-source device detection scripts and services” section.

![Figure 6-2: Download a PHP function and customize it to detect and redirect mobile visitors to your site.](image-url)
4. Develop a design that can be adapted to the needs of different devices on the fly, or create multiple designs, each optimized for a different device or device profile.

Each approach has tradeoffs, which we explore in the “Designing for Different Devices” section, later in this chapter. With either approach, expect to spend a significant amount of time creating a design (or designs) that will display well within the constraints of each device.

5. **Test, test, and test again, making adjustments as needed.**

The many variations among devices and the sheer number of devices in use on the mobile Web make it almost impossible to deliver a perfectly optimized page to every device every time. Even if you do get your mobile design strategy right, the mobile Web is a moving target. Develop a strategy that best meets the needs of your site users today (and the limitations of your time and budget), and make refinements to keep your strategy up to date in the future.

**Developing a device detection system**

The basic concept of device detection is straightforward: A system identifies the visitor’s device when it arrives at your site, and then directs the visitor to the best version of your Web site based on what the system knows about the device. (For a more technical explanation, see the upcoming sidebar “User agents and device detection.”)

The challenge with device detection — and the subsequent redirection of visitors — is that approximately 8,000 devices are in use, with more being added all the time. Although the various categories of mobile phones have similarities, the specifics of what each device supports vary.

For example, you may think that all smartphones are similar, and that within that category, all BlackBerry phones are pretty much the same. However, BlackBerry phones vary dramatically. Many BlackBerry phones have screens that are 480 pixels wide, but some are limited to 160 pixels. The latest BlackBerry phones support JavaScript, XHTML MP, and CSS2 (covered in Chapter 5), but older models provide minimal or no support for CSS. In addition, you have the human factor: BlackBerry phones offer users many options, including the ability to turn JavaScript support on and off.

How you deliver the right version of your site to the right phone is the secret sauce of many high-end mobile Web design firms. Companies such as Ansible, which developed the Microsoft Cloud Computing site featured as a case study later in this chapter, have spent years developing a finely tuned device database and device detection applications.
Chapter 6: Delivering Different Designs to Different Devices

User agents and device detection

When you open a Web page, your Web browser sends a user-agent string to the server that hosts the Web site. The user-agent string essentially introduces you to the server, describing what browser (and browser version) you’re using, as well as other things about your computer or mobile device, such as the operating system. If you want to see how your browser introduces you, go to http://whatsmyuseragent.com/.

Device detection systems are designed to recognize user-agent strings and direct visitors based on the capabilities of each device. Because many Web sites use device detection to direct mobile visitors, you can’t just open the mobile versions of some sites in a desktop browser on your computer. That’s where the User Agent Switcher comes in handy. This plug-in for Firefox (available at https://addons.mozilla.org/en-US/firefox/addon/59/) enables you to change the user agent of your Web browser, effectively tricking the server into thinking you’re using an iPhone or any of the other user agents you can load into the plug-in.

Because the User Agent Switcher makes it possible to visit the mobile version of a Web site with a browser on a desktop computer, it also makes it possible to view the source code behind those pages. This is especially valuable because browsers on mobile devices do not offer an option to view the source code, which is standard in desktop browsers such as Firefox, Safari, Chrome, and Internet Explorer.

Essentially, you build up libraries of “ways of dealing with different stuff,” explained Lee Andron, Director of Information Design and Strategy for Ansible. With every new mobile site that they develop, the company adds more processes to its object-oriented library. Thus, Ansible is constantly getting better at delivering video, audio, scripts, and other Web content to all the devices that visit their mobile sites.

You might not be able to compete with companies such as Ansible, but you can set up a similar system with far fewer resources. Like most things on the Web, open-source options are available, thanks to the generosity of programmers who share their work and offer applications for free (or for a donation). We cover a few of these options in the section that follows.

Using open-source device detection scripts and services

Although you can create your own configuration file or device database with the specifications of every mobile device you want to support and can write your own device detection scripts (if you’re an experienced programmer), you’re almost certainly better off starting with one of the open-source
options included in this section. However, even with these great resources, you’ll need some programming experience to get these solutions to work on your server.

**WURFL**

http://wurfl.sourceforge.net/

Used by a growing list of mobile Web developers, WURFL (Wireless Universal Resource File) is a freely available configuration file created by many contributors with the goal of providing a comprehensive list of all mobile devices in use today and what they can support in terms of Web design standards. Although no one claims that the list is flawless, it is regularly updated and well respected in the mobile development community.

**Tera-WURFL**

www.tera-wurfl.com

Built on WURFL (and linked to from the WURFL site), Tera-WURFL is an application created with PHP and MySQL. The Tera-WURFL application detects and matches mobile devices by first collecting user agent information from each site visitor and then passing that information to the Tera-WURFL library where it is evaluated and assigned to a UserAgentMatcher. Each UserAgentMatcher is designed to work with a device profile, made up of a group of devices with similar capabilities. Essentially, the Tera-WURFL gets you most of the way through the device detection and redirection process, but you still need to have a script that redirects visitors to the right version of your site based on the results.

**Device Atlas**

http://deviceatlas.com

If you want to create your own device database, or configuration file, a great place to start is Device Atlas. On the Device Atlas Web site you can look up the specific features supported by nearly any mobile phone, including what markup language, multimedia, and image formats each phone supports. Device Atlas also offers an API that you can use if you’re developing your own device detection solution.

**Andy Moore’s solution detects and redirects visitors**

http://detectmobilebrowsers.mobi

Andy Moore’s solution is the simplest we’ve found for setting up an auto detection and redirection solution for your site. Note, however, that you still need at least basic PHP skills to set up and install the code.
Andy Moore’s solution is a PHP function, which has eight parameters that you can easily define to handle common devices. As you see in Figure 6-3, you can use the drop-down menu options on his Web page to alter the settings for each option.

Figure 6-3: A function generator helps you customize how the PHP function directs devices on your site.

Designing for Different Devices

You can design for different mobile devices using essentially two approaches. First, you can create two or more versions of your site and set each one up at a different URL. You can think of this as a small, medium, or large approach to mobile Web design.

The second option, content adaptation, could be likened to providing a custom suit for each frame or at least enough mix-and-match options to ensure a better fit than the first option.

You can also combine these two approaches, creating multiple page designs, matching them to a general device profile, and then tailoring each design a little further, based on screen size and other factors. However, few mobile designers we’ve spoken with go to this length to produce a usable mobile experience.

We prefer the content adaptation approach, although it is the more complicated option. Following is an overview of both design strategies and how mobile Web designers use them.
Designing for content adaptation

In the content adaptation approach, you create one Web site design and then adapt it to each device. Thus, the basic site structure is the same across all devices, but specific elements, such as image size or the appearance of videos, change based on what the device can support. For example, you can optimize your images in several sizes and then deliver the version of the image that best fits the screen size of each device.

This approach is possible only if you’re working on a dynamic Web site, meaning the content of the site is stored in a database and the pages are created on the fly (as they are requested by a browser). Dynamic sites are generally built with technologies such as PHP, ASP.NET, or Java combined with a database like MySQL. An example of a dynamic Web site is Amazon, which creates a page that recommends books for you that are different from the ones it recommends for others, based on your previous visits to the site and books that you have purchased. A sophisticated dynamic site such as Amazon works by retrieving the images and descriptions of the books from a database and assembles them on a Web page as you access the site.

Many developers who follow the content adaptation approach also use caching. That means they can save, or cache, a version of the site optimized for a particular kind of phone. For example, if 40 percent of your traffic comes from iPhones, and all iPhones have the same specifications, you can save a lot of server power by creating the iPhone version once and then caching it so that you can serve that cached version rather than generate a new version for every iPhone that comes along.

The content adaptation approach is complicated and only suited to dynamic Web sites. If you’re working on a static Web site, where each page of the site is a separate HTML file, your best option is to create a few different page designs and set each one up at its own URL, which is the approach covered in the next section.

Creating mobile versions of a blog

Web blogs created with WordPress (or other blogging software) are a kind of dynamic Web site. All headlines, posts, and images are stored in a MySQL database, and pages are generated on the fly. Thus, creating a mobile-optimized version of a blog is relatively easy, and there is an ever-expanding list of blog themes available on the WordPress Web site to help you automate the process. Some mobile-optimized WordPress themes even take care of device detection, including the one featured in Chapter 9.
Chapter 6: Delivering Different Designs to Different Devices

Creating different versions of a site

Creating different versions of a Web site is easier to implement than the content adaptation approach, but you won’t be able to match your designs to the unique specifications of every device as effectively. Still, creating different site versions is far superior to creating just one mobile version of your site (an approach covered in Chapter 3).

Because it’s impractical (if not impossible) to create a different design for each of the thousands of devices that might visit your site, most developers start by creating two or more device profiles — categories of phones with similar features.

Here’s an example of the two main profile categories you should consider, followed by a more detailed breakdown for low-end phones:

**Profile 1:** A device profile for high-end touch-screen phones would include the iPhone, phones that run the Android operating system, those with the latest Palm OS, and the newest phones with the Microsoft Internet Explorer Web browser. With these as your target, you can design a version of your site optimized for a 320-pixel-wide screen, that includes JavaScript, video, and the latest markup language HTML5, as well as CSS3 (covered in Chapter 5). Because these high-end phones all use Web browsers that support this technology, you can do a lot with them.

**Profile 2:** A device profile for low-end phones would include older BlackBerry and other smartphones, as well as all feature phones. In this profile, you would leave out JavaScript, which is not supported consistently (or at all) across these phones, limit or remove any video or audio files, and create the page markup with XHTML MP and CSS MP (covered in Chapter 5).

You can create as many profiles as you want and can base the profiles on any number of factors, but these two broad categories will cover most phones today. If you want to better serve the low-end phones, you could then create a second level of device profiles based on screen size.

If you create device profiles based on screen size, and you make them a subset of the first pair of profiles, you might define them as follows:

**Profile 2A:** Tiny screens, those limited to 132 pixels or less

**Profile 2B:** Small screens, between 132 and 240 pixels

**Profile 2C:** Medium screens, those between 240 and 320 pixels
Part II: Following Mobile Web Standards

128

As you create each mobile version of your Web site, consider how to take best advantage of the capabilities of the phones in each profile. For example, you may want to include video in the version you design for the Profile 1 phones, but not in the Profile 2 version you create for feature phones, which don’t support video as well.

After you’ve built the different versions, you can publish each one at a special URL. For example, you could publish the site designed for Profile 1 at smartphones.yourdomainname.com and then publish the Profile 2 version at http:m.yourdomainname.com because it’s more difficult to type on a feature phone. Also note that although you can use a domain name such as yourdomainname.com/mobile, instead, for a feature phone, it’s best to use the m.domain example because it’s difficult to add a forward slash on most feature phones. If you then go on to the second level of profiles, you could create three variations on that domain, one for each screen size.

With separate domains set up for each profile, you can then direct visitors to the best version using a simple detection script, like the one available on Andy Moore’s Web site, covered previously (in the “Using open-source device detection scripts and services” section).

Planning a Mobile Web Project

Whether you will be creating multiple versions of a site or using content adaptation, it’s wise to begin by thinking about what you want to include on the site and how to design the site so that it works on a wide range of devices.
To help you understand some of the best practices in planning a mobile Web site, this section focuses on a real-world mobile design success story from Ansible, Interpublic Group’s full service mobile marketing agency.

The case study is based on a project that the Ansible team developed for Microsoft: transforming the desktop version of the Cloud Computing site, shown in Figure 6-4, into a mobile version that displays well on nearly every mobile device that might visit the site, including the Motorola RAZR, Apple iPhone, and BlackBerry Bold, also shown in Figure 6-4. Lee Andron, director of information design and strategy, and Sia Ea, senior creative manager, generously granted us a series of interviews. In addition, the Ansible team provided access to planning and design documents.

For the Cloud Computing site, Andron created the wireframe and the final Project Planning Document, Ea created the design, a team of programmers did the development work, and a project manager oversaw the entire process.

Figure 6-4: The Microsoft Cloud Computing site as it appears in the Firefox Web browser on a desktop computer.
Creating wireframes

On larger projects such as the Cloud Computing site, Creative Director Andron explained, it’s good practice to start by creating a wireframe, a skeletal model of a site not unlike a blueprint for a building. Wireframes are not unique to mobile design; most design firms that build big, complex Web sites create wireframes.

A good wireframe is like a diagram of a site that shows what happens when someone clicks on each page. Most developers who create wireframes agree that it’s important to not add design elements at the initial stages — no color, no fancy fonts. The wireframe is a structural document designed to help you focus on how a site works without being distracted by design details, such as the color scheme. As the development process evolves, the initial wireframe document may grow to include design mockups and other details.

In the top-right corner of each wireframe document (such as the one in Figure 6-5), Andron included a miniature site map that shows where the page fits with the rest of the site.

Figure 6-5:
The initial wireframe document shows only the bare-bones structure of each page.
Although beginning by creating a wireframe may seem like it delays the start of development, most Web designers find that the project is more efficient in the long run because the design team can work out potential problems with navigation and functionality before adding the complexity of programming and design. And after the wireframe is approved, you can move forward with technology and design simultaneously because everyone is working from the same blueprint.

Dividing a project into manageable phases also makes it easier for clients to approve each stage of development; each phase can have a distinct focus, Andron said. Our client approves functionality, then design, and finally the text used on each page. Each approval is documented in the growing Project Design Document, which serves as a set of blueprints in the development process.

### Resources and software for creating wireframes

You can create a wireframe document in almost any software program. Some designers use common applications, including Microsoft Word or PowerPoint, but there are a number of programs and online services designed specifically for creating wireframes. Here are a few:

**Cacoo** ([www.cacoo.com](http://www.cacoo.com)): This online drawing program includes many icons and other features designed for creating wireframes, sitemaps, and charts. (Cost: free)

**Balsamiq Mockups** ([www.balsamiq.com](http://www.balsamiq.com)): This online drawing and collaboration tool offers more sophisticated features than Cacoo. (Cost: $79)

**iPlotz** ([www.iplotz.com](http://www.iplotz.com)): An online tool you can use to create clickable and navigable wireframes and prototypes. (Cost: $15/mo)

**Fairbuilder** ([www.flairbuilder.com](http://www.flairbuilder.com)): This online tool can be used to create complete interactive prototypes and wireframes. (Cost: $24/mo)

**Microsoft Visio** ([www.microsoft.com/visio](http://www.microsoft.com/visio)): A popular program among professionals who favor Microsoft products, Visio is only available for Windows computers. The program uses vector graphics and features a broad collection of templates and other sophisticated tools designed for developing complex projects. (Cost: $500+ for the professional version)

**Omnigraffle** ([www.omnigroup.com](http://www.omnigroup.com)): This program works on Macintosh computers and iPads and can be used to create diagrams, wireframes, and charts with many great design features built in. Omnigraffle can also import and export Visio documents. (Cost: $200 for the professional version)


Stepping through the development process

At Ansible, each new project begins with a Vision Document that defines the concept of what the site will accomplish. It helps the firm come to an agreement with the customer about the goals the site needs to achieve as well as the expected return on investment, or ROI. Andron is quick to admit that just because you can develop a different design for every mobile device, doesn’t mean doing so is the best investment for all of his clients. As with any design project, you have to balance the options with the expected return on investment.

Even before Ansible signs a contract with a client, Andron often creates an initial site map. The initial site map often changes quite a bit by the time they get to the final version of the wireframe document (shown in Figure 6-6), but it serves as a guide that can help make sure the client and the development team are on the same page in terms of the scope of the project.

After a client signs off on the initial site map document, Andron and his team create a statement of work, where they define a budget and timeline.

Figure 6-6:
A site map in the wireframe shows the project’s scope and how each page links.
Once the client signs a contract, the team is ready to move full steam ahead. The entire development process goes something like this:

1. Create the detailed wireframe designs complete with site maps (refer to Figures 6-5 and 6-6) and discuss with the client to get approval.

2. The creative designer creates the initial page designs and presents them to the client for approval. For this project, the senior creative manager, Sia Ea, created designs for each of the main pages of the site, including the one shown in Figure 6-7.
3. Add the final copy — the text that will appear in the pages of the site, and get the client to sign off on all content.

The team at Ansible, like many designers, often uses “greeked” or placeholder text to show how copy will look on the initial page designs. However, Andron prefers to use real content from the site whenever possible because it can help clients more clearly visualize how the site will come together in the end. Screen designs are so small that text plays a fundamental roll in the design. He says, “If we include text in the site sourced from the campaign, we save time in the long run by suggesting what the copy of the site should say to achieve the goals of the project.”

4. Combine all the elements (wireframe, design mockups, and content) into one complete Project Design Document, like the one shown in Figure 6-8.

![Figure 6-8: The middle of the page shows two kinds of navigation, one for touchscreen phones and one for keypads on a feature phone.](image)

Creating a wireframe helps you work out complex navigational challenges before you get to the programming and development of a site. For example, the middle section of Figure 6-8 shows two kinds of navigation that will appear on the home page, one for devices with touch screens and a second for feature phones that are limited to numeric keypads. The need for dual navigation options makes mobile sites complex, but working out how to handle these differences before you start programming helps the development go more smoothly.
5. Notes are added to the document throughout the process.

At each stage of review with the client, Andron adds notes from the client to the Project Planning Document and includes those notes when the final document is passed on to the programmers. One of the challenges with creating a wireframe and other parts of the planning document, he explained, is that you’re really serving two masters — one is the customer because the document must get their approval, and the other is the development team because, ultimately, the document has to be useful to the programmers and designers who create the site.

6. After the wireframes, creative designs, and copy deck are approved, the Project Design Document is complete and the programmers put the plan into action.

7. The final phases before the site goes live involve lots of testing.

Andron said his team uses services like DeviceAnywhere to help with initial testing because it offers so many different devices on one site and the ability to test mobile services in other countries without having to physically go there. “The best way to test a mobile design, however, is to use real mobile devices,” he said. In his office, Andron keeps several mobile phones on hand and uses them to view pages and test interactive features throughout the development process. (You find more about DeviceAnywhere and other testing options in Chapter 7.)

After the site is built on a testing server and ready for final review, Andron demonstrates the site for the client on at least five phones running different operating systems and browsers (depending on the client’s target audience).

“Developing mobile Web design effectively requires a constant improvement process,” Andron said, noting that at every stage, they are testing their assumptions and making changes as necessary to ensure that the final project will be successful.

The entire process of developing a Project Design Document with planning documents, such as the ones shown in Figures 6-5 through 6-8, may require more than a dozen revisions as new elements are added and changes are requested. The Ansible team created 14 versions of the Microsoft Cloud Computing site Project Design Document in the process of developing a site to reach the project goals. The final version consisted of an 18-page document, complete with a site map, navigation documents, real content, and design mockups plus the copy deck.

Andron saves a copy of the wireframe document with each significant revision so he can backtrack if things change significantly along the way, which is a common challenge with projects that include so many moving pieces. “We might have six or seven revisions before we even show the wireframe to a client,” he said.
Considering the designer’s perspective

The constraints that designers face when creating sites for the mobile Web would leave many artists feeling as though they were being asked to build three different model ships in one tiny bottle, while wearing a straightjacket. But Sia Ea, who was the design lead on the Ansible team for the Microsoft Cloud Computing site project, enjoys the challenge.

“Creating a Web design that looks good and works across a wide variety of mobile phones and other devices isn’t easy,” she said, “but it is satisfying.” Ea likes to start by creating a version of the design that includes all of the most advanced features possible, and then stripping out the elements that won’t work on low-end devices as she creates different versions.

Because the development process begins with a wireframe, her options are limited from the start by more than just the constraints of the mobile Web. “The wireframe is a map for the site, and includes many assumptions that also affect the design,” she said. Although she had to base her design on the wireframe, she was able to negotiate some adjustments with Andron before they presented the design to the client.

Ea kept the colors already in use on the desktop version of the Cloud Computing site (refer to Figure 6-4), which consisted of bright colors on a white background. Fortunately, this color scheme makes text easy to read on a mobile device as well as on the desktop site. That said, Ea prefers to use darker backgrounds, with a light text color for navigation links because it helps them stand out on a small screen and makes it easier to identify where a user must click on a link (see the bottom of Figure 6-9 for an example).

Ea thinks anyone who has been designing for the desktop Web has a great background for mobile design because they already have experience designing for different computer platforms and browser differences. But they may still have to learn to deal with the fact that the differences among mobile devices are much greater because there are so many different devices and resolutions. In addition, browsers have different capabilities; for example, some can’t render tables, and others don’t render CSS or JavaScript. Knowing these limitations is critical when creating a design that will work across many devices.

Now the iPad is adding new challenges and opportunities in mobile design. In the past, the largest design Ea would create was 320 pixels wide, for an iPhone. As mobile screens got larger, she started making the largest images 480 pixels wide. Now she makes graphics that are 768 pixels wide so that they fill the screen on an iPad. In general, Ea creates the design with images in the largest size and then scales down the design for smaller devices.
Following are some of Ea’s tips for mobile Web design:

✓ Stick with fundamental design principles. An element’s typography, composition, location, and scale are all important.

✓ Use grids for layout, and use colors that provide contrast. Use a white background and dark text when you have lots of copy, and use the reverse for site imagery. Keep links simple and easy to navigate.

✓ Don’t make pages too long (no more than two or three screens). Keep copy short and ensure that the most important information appears at the top of the screen. “Having an iPhone will spoil you,” she said, “because scrolling down a page on many devices requires tediously pressing a key, so limiting page length is extremely important.”

✓ Keep users entertained and give them the capability of moving forward or backward.

Ea is quick to admit there are exceptions to any design rule. Although she generally recommends that you limit each page to no more than two screens of content, Figure 6-10 shows a page in the Cloud Computing site that is
longer. The page shown in Figure 6-10 lists all the RSS feeds to which users of the Cloud Computing site can subscribe. “The desire to show the extensive list of articles users could choose from justified creating a longer page layout in this case,” she said.

Figure 6-10:
Page designs were limited to no more than two to three screens of content, except when showcasing a long list of RSS feeds.