Developing Mobile Wireless Applications

Kris Read and Frank Maurer • *University of Calgary*

When we talk about Web content, we typically use document metaphors — saving our places with bookmarks, viewing and downloading pages, and so on. This metaphor feels natural because it originated in Web content, which is basically a collection of electronic documents laid out in a format familiar from books, magazines, and other publications.

In the mobile world, however, developers quickly realized that "page" was not the best content metaphor. Small screens, limited bandwidth, and relatively little storage space prevent us from building large, complex page layouts. We thus prefer to think of our content as a deck of cards. A card, like a cell-phone or PDA, is small and might fit into the palm of your hand, for example. Also, like mobile device screens, individual cards offer relatively little space to write on — though you can write a little on each card and flip between them whenever necessary.

This deck-of-cards metaphor underlies nearly all content developed for mobile wireless technologies. Standard, high-level application-development technologies, which are available for most major hardware platforms, ease this development task because they're optimized for the severe restrictions on a mobile device's screen size, processing power, and battery life. Here, we discuss three major application-development technologies: the wireless application protocol (WAP), NTT DoCoMo's i-mode, and Sun's Java 2 Micro Edition (J2ME). Each lets developers create content for the current and upcoming generation of mobile wireless devices.

The Wireless Application Protocol

WAP (www.wapforum.org) is a free, open specification that lets wireless devices easily interact with services and each other. It works over all major wireless networks and on all major mobile operating systems. In fact, renowned wireless technology providers such as Ericsson, Nokia, and Motorola are endorsing WAP for their 2.5G and 3G networks. Microsoft is also backing this mobile solution.

Developers were slow to accept early WAP versions because of a steep learning curve and incompatibility with existing Web content. WAP 2.0 adopts existing Web standards, however, and specifies how developers can best use them in a wireless environment. WAP 2.0's aim is to improve the user experience as wireless networks slowly expand to allow richer content and cheaper bandwidth.

Basic Features

In theory, WAP 2.0 lets developers create applications that feature animation, streaming media, and music downloads. WAP 2.0 displays color graphics, provides location-specific content, and allows synchronization with a remote desktop PC. The goal is to reduce development costs by letting developers use standard tools to write applications for Web browsers and WAP clients at the same time.

In WAP 2.0, developers write content in XHTML using the XHTML Basic profile. (As the "XHTML Fundamentals" sidebar explains, XHTML is quite easy to produce if you're familiar with existing Web development technologies.) WAP 2.0 also supports cascading style sheets (CSS) for document formatting. Users view XHTML content through a client device's microbrowser; content is either stored locally or served up by XHTML-compatible Web sites. Not all existing HTML Web pages can be viewed with this technology, however, and to make a Web site WAP-2.0 compatible, developers must follow

XHTML Fundamentals

The Extensible HyperText Markup Language is the W3C's relatively new standardized Web document format. XHTML is based on XML and is ultimately meant to replace HTML as the accepted Web page format.

XHTML returns document formatting to HTML's original intentions by separating content and format. Because of browser platforms' increasing range — from desktop to television set-top boxes and mobile devices — the W3C has defined XHTML subsets, or *profiles*. WAP 2.0 developers use the XHTML Basic profile, which is a subset of all available XHTML features. As the W3C states, it includes "the minimal set of modules required to be an XHTML host language document type," including images, forms, basic tables, and object support. This might seem like a step backward from HTML 4.0, but in fact some of its features — such as frames or complex tables — are simply inappropriate for small devices. Developers can use XHTML Basic to create fully featured Web sites for ordinary Web browsers, but it's also simple enough to scale down to a mobile device's limited resources. There is even a tool, HTML Tidy, which helps convert HTML to XHTML; it's available at http://tidy.sourceforge.net.

specific XHTML guidelines. Currently, few Web sites are written in XHTML, but the number should grow rapidly as developers and tools conform to World Wide Web Consortium (W3C) standards.

WAP 2.0 includes multimedia messaging service (MMS), which lets users send messages that include images and text. WAP supporters expect MMS to further popularize short message service (SMS) in Europe and North America. WAP 2.0 also includes WAP Push, which automatically delivers content to users, rather than waiting for them to request it. This is useful for services such as online auctions or stock trading, where it's important for users to receive information the moment something interesting happens.

WAP 2.0 is backward compatible with the previous standard, WAP 1.x, which uses Wireless Markup Language (WML) rather than XHTML for document formatting. WML is a set of specific XML markup tags the WAP forum created specifically for WAP to represent typical mobile wireless content. WML's main disadvantage was that developers had to create content independently of HTML, or use some sort of translation process. Although some of WML's tags are similar to HTML, the document elements are different and the feature set is much smaller. Compare this example from WAP 2.0 XHTML Basic:

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML
         Basic 1.0//EN"
 "http://www.w3.org/TR/xhtml-basic/xhtml-
basic10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

   <head>
      <title>My Basic Page</title>
      <style type="text/css">
               a { color: #0066FF }
         </style>
      </head>
      <body>
         <img src="icon.gif"/>
         <p>Here is some text.</p>
         <a href="index.xhtml">A Link</a>
      </body>
   </html>
```

with this example from WAP 1.x WML:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "//WAPFORUM//DTD 1.1//EN"
  "http://www.waponline.com/wml_1.1.xml">

<wml>
   <card title="My Basic Card">
      <p>
      <img src="icon.wbmp" alt="icon"/><br/>
      This is some text.<br/>
      <anchor>A Link.
      <go href="index.wml"/>
      </anchor>
      </p>
   </card>
</wml>
```

In the XHTML Basic example, we define an HTML document with two parts: `head` and `body`. The `head` includes at least a title, but here we also show how XHTML Basic supports CSS. The `body` uses familiar HTML tags such as paragraph (`p`), image (`img`), and anchor (`a`). In the second example, we define a WML document rather than an HTML document. Then, according to the deck-of-cards metaphor, we must define a series of cards. In this example, we define only one: it inclu1des an image (`img`), a paragraph (`p`), and an `anchor`. Figure 1 shows a WAP 1.x document as viewed using 3tL's free WAP Emulator, which lets PC users preview WAP documents without using a mobile device. Although WML's text syntax and image components are much like HTML, the `anchor` is not. When using WML, you can't take HTML elements for granted; you must ensure that any tags you use are part of the WAP Forum WML 1.x specification (available at www.wapforum.org/what/technical.htm).

Unlike WML, XHTML Basic can be displayed in the latest major browser versions, including Internet Explorer, Mozilla, Netscape, and Opera. However, it does have some small eccentricities that you should keep in mind. Unlike HTML pages, for example, XHTML pages must include a `<title>` element. For mobile use, you should keep the title length rather short. In XHTML Basic, you can also use some unconventional parameters to make your mobile application friendlier. One feature likely to be popular, for example, is the *accesskey* parameter:

```
<a href="next.html" accesskey="1">Next</a>
```

In this case, the accesskey parameter lets users press the "1" on their mobile keypad to click the `Next` link. Such shortcuts are smart and reasonably universal in their compatibility with device hardware; if your browser doesn't support the accesskey parameter, it will simply ignore it. There are many other mobile-friendly parameters and tags in XHTML Basic; a complete listing is available at www.w3.org/TR/xhtml-basic.

Status and Outlook

Books and online tutorials about WAP 2.0 development are already starting to appear, but an experienced Web developer can probably leap in without assistance. Although most WAP service providers are still using WAP 1.x, version 2.0's backward compatibility should lead to relatively rapid upgrades. These upgrades also make sense in that industry is moving toward adopting XML-based formats for almost every kind of data interchange. WAP 2.0 makes it easy to adapt existing knowledge to the mobile arena, and will thus gain rapid industry acceptance. For more information about WAP, visit the W3C (www.w3.org/TR/NOTE-WAP) or WAP Forum (www.wapforum.org).

i-mode

Japan's NTT DoCoMo (www.nttdocomo.com/top.html) first offered the wireless i-mode Internet service in February 1999. Currently, i-mode covers almost all of Japan, with 32 million existing subscribers and about 30,000 more added per day as of March 2002. Parts of Europe have also adopted i-mode because of its strong business model. A large base of i-mode content is available, and several hardware devices support it as well.

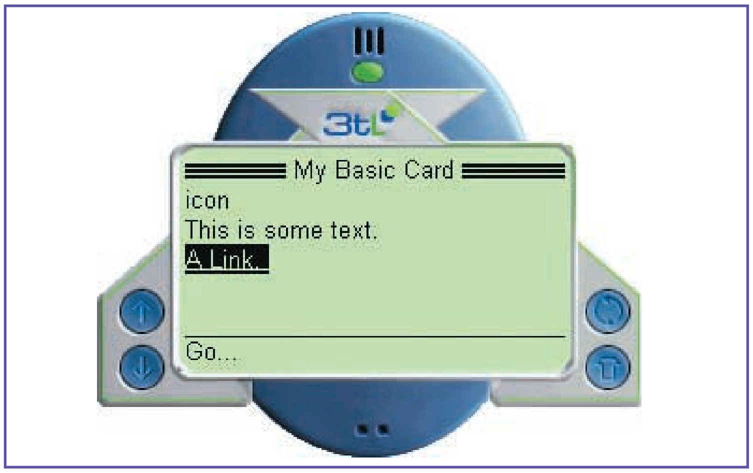

Figure 1. A Wireless Access Protocol 1.x document as it appears on the 3tL WAP Emulator. WAP 1.x uses the Wireless Markup Language rather than XHTML to format documents.

Because WAP is a protocol and i-mode is a service, the two technologies are not necessarily competitors; DoCoMo is a member of the WAP Forum. In fact, according to WAP Forum's CEO Scott Goldman, i-mode and WAP will likely converge sometime in 2003 as both move toward W3C standards support and seek the widest possible range of hardware vendors.

Basic Features

You can use i-mode to reserve tickets, find a restaurant, check your bank balance, transfer money, pay bills, read news reports, get the weather forecast, view train schedules and city maps, and more. You can send and receive e-mail on i-mode phones, both to other users and to any Internet address. Furthermore, i-mode phones can access the Internet directly; users can browse to any i-mode compatible URL, rather than being limited to provider-specific content. Unlike a typical cellular service, i-mode charges users according to the volume of data transmitted, rather than total connection time. This business model is highly appealing to users because it lets you stay connected to a single Web site for hours without a fee, as long as you don't transmit data.

When i-mode was launched in 1999, there was no suitable standard for compact document formatting. NTT DoCoMo thus created compact HTML (cHTML), i-mode's content development language. In general, cHTML is a smaller, slightly modified version of HTML — optimized for low bandwidth and limited client resources. Among the HTML 4.0 features that cHTML excludes are JPEG images,

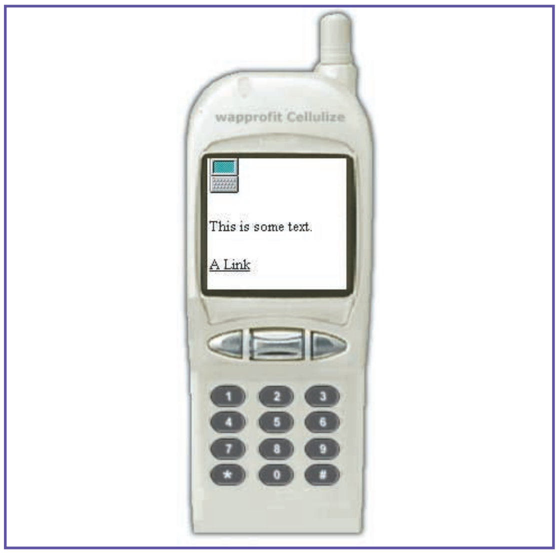

Figure 2. cHTML example inside wapprofit's i-mode Phone Emulator. In cHTML, documents are structured like HTML documents.

tables, image maps, character fonts and styles, background colors and images, and frames.

Like WML, cHTML's main disadvantage is that it's not a W3C standard. Unlike WML, however, cHTML is similar enough to HTML to be generally easy to create, and is viewable using common Web browsers, such as Internet Explorer. In many ways, cHTML was XHTML Basic's predecessor, though it doesn't support CSS.

Developing a cHTML page is relatively simple. Figure 2 shows the following code example as it appears inside wapprofit's i-mode Phone Emulator.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD
      Compact HTML 1.0 Draft//EN">
   <HTML>
      <HEAD>
      <TITLE>My cHTML Page</TITLE>
      </HEAD>
      <BODY>
         <IMG SRC="icon.gif">
         <P>This is some text.</P>
         <A HREF="index.html">A Link</A>
      </BODY>
   </HTML>
```

Although it has fewer features, cHTML is much like HTML. Its documents are structured like HTML documents, with a `head` and `body`, and `paragraph`, `image`, and `anchor` tags, as in the XHTML example; cHTML also has its own parameters and tags. Along with the `accesskey` parameter (which we borrowed in our XHTML example), a cHTML document might have the following anchor:

```
<A href="tel:555-0123">Call Frank</A>
```

With this link, a user can open a voice connection to the telephone number provided from a cHTML Web page. The cHTML specification contains many other additions to HTML, but all are optional.

Status and Outlook

In mid 2002, NTT DoCoMo announced plans to develop new 3G handsets equipped with a dual browser capable of handling both i-mode cHTML and XHTML Basic. The company was scheduled to introduce the handsets in some locations by the end of 2002. This move sets a precedent for WAP migration, but also helps prevent the isolation of existing cHTML developers and content.

The i-mode service is immensely popular and has a large user base. Given that an estimated 81 percent of the world's wireless Internet use is in Japan, i-mode's impact on future technologies and mobile development is likely to be quite significant.

Java 2 Micro Edition

J2ME (http://java.sun.com/j2me/) is Sun Microsystems' contribution to the wireless field. However, J2ME is not designed specifically for wireless applications; Sun advertises J2ME as a Java platform for all kinds of consumer and embedded devices, from phones to set-top boxes.

Like regular Java, J2ME applications are portable, and are therefore available on anything that runs a Java Virtual Machine. While XHTML and cHTML are markup languages that require browser software on the client device, J2ME provides a complete application-programming framework for mobile devices. A browser interface, such as those used by WAP or i-mode, is convenient for accessing static content or displaying a list of applications and services, but it's not suitable for interacting with users. J2ME gives users control over their interface and direct access to their hardware platform, with built-in secure networking capabilities and customizable input features.

Basic Features

J2ME applies to wireless devices through its Mobile Information Device Profile functionality. According to Sun, MIDP in its Connected Limited Device Configuration (CLDC), "is the Java runtime environment for today's mobile information devices."

MIDP lets developers create small applications called *Midlets*, which can include games, instant messaging, e-mail, financial applications, and vendor services. J2ME Midlets are to wireless what

Applets have been to the Web; Midlets let you develop a richer, more interactive client experience that is portable, downloadable, and robust.

The Midlets' disadvantage is that, like Applets, they consume considerable storage and memory space, and the initial download can take a long time. Users are more likely to repeatedly use familiar services on mobile devices, however, so J2ME is a good fit for most mobile application needs.

To develop J2ME Midlets, you can download the J2ME Wireless Toolkit (http://java.sun.com/products/j2mewtoolkit), which provides documentation and examples, as well as an emulation environment for simulating a CLDC/MIDP-compliant mobile phone or PDA on your workstation. You can then develop Java applications locally and test them on the Sun-approved emulator; you don't even have to own a phone to develop phone applications. Following is a simple Midlet application. Figure 3 shows it running on the emulator's DefaultColorPhone.

```
import javax.microedition.lcdui.*;
import javax.microedition.midlet.*;
public class HelloMIDlet
    extends MIDlet
    implements CommandListener {

  private Form mMainForm;

  public HelloMIDlet() {
    mMainForm = new Form("HelloMIDlet");
    mMainForm.append(new
      StringItem(null, "Hello World!"));
    mMainForm.addCommand(new
      Command("Exit", Command.EXIT, 0));
    mMainForm.setCommandListener(this);
  }

  public void startApp() {
    Display.getDisplay(this).
      setCurrent(mMainForm);
  }

  public void pauseApp() {}

  public         void         destroyApp
    (boolean unconditional) {}

  public  void  commandAction(Command  c,
Displayable s) {
    notifyDestroyed();
  }
}  //  example  from  http://wireless.
    java.sun.com
```

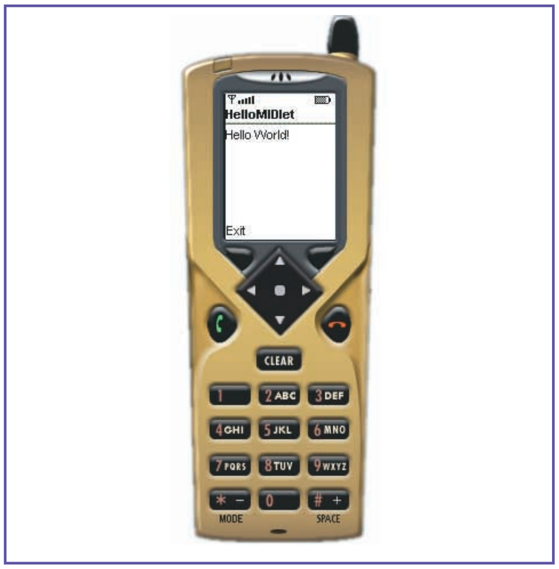

Figure 3. Example J2ME application running on the DefaultColorPhone emulator. The emulator is included in Sun's J2ME Toolkit.

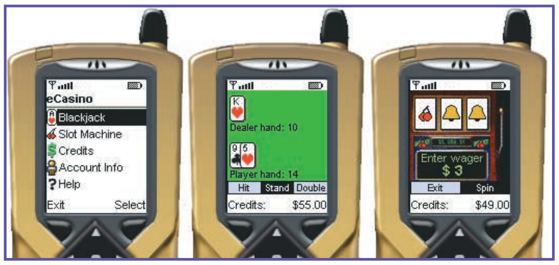

Figure 4. A game developed using J2ME. This secure client-server MIDP casino game, developed by University of Calgary students, features both online blackjack and a slot machine, and simulates debiting and crediting an actual account.

As you can see, the Midlet is slightly more complicated than developing a page using cHTML or XHTML, but it has the potential to do a lot more. Returning to the deck-of-cards metaphor, with Midlets you often have to implement many views or "cards," then choose which one the system will present "face up" for the user at a particular time. Figure 4 shows a casino game developed using J2ME.

In our code example, we extended the `Midlet` class, which provides the basic methods for interacting with our device. We also implemented the `CommandListener` interface to handle input and output. Our constructor created a new instance of `Form`, one of J2ME's many useful "card" objects. Our `Form` lets us interact with users much like a Web-based form, using input boxes, buttons, checkboxes, and so forth. In this case, all we do is print out the string `Hello World`. Elements

Wireless Development Resources

WAP
- Nokia WAP Home Page—www.nokia.com/wap
- Jataayu WAP FAQ—www.jataayusoft.com/WAP2-FAQs.pdf
- WebReference XHTML resources—www.webreference.com/authoring/languages/xhtml/
- Silicon.com's Guide to WAP—www.silicon.com/siliconguides/wap/wap.htm
- International Engineering Consortium—www.iec.org/online/tutorials/wap/
- Wireless Developer Network—www.wirelessdevnet.com
- WAPDrive—www.wapdrive.com

i-mode
- Time for i-mode—www.corrine.nl/
- Addison-Wesley's imode guide—www.imode-guide.com/resources/
- Mobile Media Japan—www.mobilemediajapan.com/
- palowireless i-mode Resource Center—www.palowireless.com/imode/
- XML.com: Getting into i-mode www.xml.com/pub/a/2000/09/20/wireless/imode.html
- Eurotechnology i-mode FAQ—www.eurotechnology.com/imode/faq.html
- i-mode links FAQ—imodelinks.com/desktop/faq.html

J2ME
- Java Wireless Developer Homepage—http://wireless.java.sun.com
- The Micro Java Network—www.microjava.com
- OnJava J2ME/Wireless—www.onjava.com/onjava/wireless/
- Core J2ME—www.corej2me.com/
- JGuru J2ME FAQ—www.jguru.com/faq/J2ME
- The Lurker's Guide to J2ME—www.blueboard.com/j2me/
- J2ME Advisor—www.advisor.com/adv/J2MEAdvisor
- J2ME.org—www.j2me.org/

added to a `Form` object appear in the order that we add them, so in this case our string would be at the top. We must also bind the `Exit` command so that our application actually does something. We create new commands with a description ("exit"), a button preference (`Command.EXIT` refers to the default exit button available on most devices), and a command ID (which is user-defined). The Midlet uses the command ID as a numerical reference that indicates which command a user has selected.

When a user starts our program, the `startApp` method is called to show the first card by setting the display to show our `Form` instance. The `commandAction method` is called whenever a command is detected; because our only command is to quit the application, our example method doesn't require conditional logic. In reality, you might have many commands, such as `Next`, `Previous`, `Quit`, or `Confirm`, and you'd have to implement the command logic within the `commandAction` method.

Status and Outlook
Because it has so much available functionality, J2ME is best used for complicated or interactive applications, or when WAP and i-mode are unsuitable or unavailable. In the US, many companies, including Sprint, offer data services using applications developed specifically for J2ME, but you can also run J2ME in conjunction with WAP- or i-mode-based services. MIDP 1.0 includes APIs for application lifecycle, HTTP network connectivity, user interface, and persistent storage. Many MIDP 1.0 applications already exist. The newly released MIDP 2.0 includes many enhancements and additions, including scheduling, secure networking, and both multimedia and game APIs.

More information on creating Midlets, along with example code for more complicated Midlets, is available at http://wireless.java.sun.com.

Looking Ahead
As the popularity of mobile technology continues to rise and its costs continue to fall, demand for Web and other services over mobile devices will increase. It's crucial, however, that content providers continue to offer services that are useful and that customers will pay for. Idyllically, what we need is the infamous killer app: an application that has such value to customers that it will motivate their investment in relatively costly wireless mobile devices. Such a killer app has yet to be found, so it remains to be seen if the value of mobile wireless applications outweighs the price. Businesses also need to think about a model for mobile services or mobile commerce that is profitable and practical, and that fits with available technology. In any case, there is already demand for knowledge about the technologies described here, and it's easy to see a potential for future growth.

Kris Read is a graduate student in the Computer Science Department at the University of Calgary, where he specializes in software engineering. Contact him at kdread@ucalgary.ca.

Frank Maurer is a research professor in software engineering at the University of Calgary. His research interests include agile software processes, experience management and Web technologies. Contact him at maurer@ucalgary.ca.