16

Mobile Device Development with ASP.NET

MOBILE DEVICES ARE BECOMING INCREASINGLY popular. Over the past few years, mobile phones have become so common that it is more of a rarity to see someone without one. *Personal Digital Assistants* (PDAs), such as Palm Pilots, also are popular. Combine this with the fact that the Internet is everywhere. It goes without saying that tremendous opportunity exists for the savvy web developer to take advantage of this current state of accessibility.

So, how can a developer take advantage of this? Consider the following: Most people use the Internet as a means of finding information. Ironically, it is not always possible to get to an Internet PC or Mac; however, most people have more convenient access to a mobile phone or PDA. This means that there is an enormous, untapped market for mobile Internet services. After all, imagine being able to use your mobile device not only for sending and receiving your email, but also for receiving information from the sports and news feeds or the company you work for, or being able to book and purchase tickets through online services. The possibilities are limitless!

This chapter introduces the mobile device types you can develop, and provides a brief overview of the underlying technology used for writing applications for mobile devices:

- Covering Wireless Application Protocol (WAP)
- Wireless Markup Language (WML)
- WMLScript (the script language for WML)

This chapter provides an overview of the ASP.NET Mobile Internet Toolkit (SDK designed for developing Mobile Device applications). It also explains some issues dealing with mobile device development and some key differences between ASP.NET forms and controls, and ASP.NET Mobile forms and Controls. From here, it describes each Mobile control and gives examples of their uses. Finally, a brief overview of the mobile device support in the application presented in Chapter 17, "Putting It All Together" is provided.

Software You Need for This Chapter

To use the examples in this chapter, your development machine has to have the following software installed as well as ASP.NET:

- **Mobile Internet Toolkit**—This can be downloaded from `http://www.asp.netwww.asp.net`.
- **A WAP-enabled Microbrowser or emulator**—this book uses Openwave UP Simulator 4.1. For the examples in this book, you can download this emulator from `http://www.phone.com`.

Wireless Application Protocol (WAP)

The *WAP* architecture is not that different from the WWW architecture. In fact, the WAP architecture is based on the existing WWW architecture, which means if you understand the WWW architecture, you can understand the WAP architecture.

Most of the technology developed for the Internet has been designed with the desktop user in mind, which in itself presents some rather interesting issues when developing Mobile Internet Device applications. For instance, a desktop user has a large display with which to view, a keyboard for data entry, and a fast Internet connection. Compare this to the mobile device user who has a limited display area and limited data entry ability.

The WAP architecture, although based on existing web technology, has numerous optimizations for wireless data transfer. Most of these optimizations deal with the fact that the wireless data communications technology available to the public has a small bandwidth capacity. In most cases, the bandwidth capacity is less than 15Kbps, which is considerably less when compared to conventional web browsing technology, which runs at an average minimum of 56Kbps.

When a mobile device user requests a web page, the following request and response process occurs:

1. The user requests a URL from his Microbrowser.

2. The WAP browser encodes the request into WML format and then sends the request to a WAP gateway.
3. The WAP gateway receives the WAP request, converts the WAP request into an HTTP request, and then sends it to a web server.
4. The web server receives the HTTP request, performs whatever processing is required, and then sends back an HTTP response to the WAP gateway.
5. The WAP gateway receives the HTTP response, converts the HTTP response into a WAP response, and then sends it to the WAP device that requested it.
6. The WAP Microbrowser software receives a WAP response and renders it to the mobile device display.

WAP Forum

The *WAP Forum* is an association that developed the WAP standard. It is made up of more than 500 members worldwide. It has the sole objective of promoting the WAP standard and assisting companies in the adoption of the standard.

For more detailed information on the WAP architecture, go to the WAP Forum web site at www.wapforum.org.

Wireless Markup Language (WML)

The WAP architecture also includes a markup language that is similar to HTML in structure; this is called *WML*. This markup language is used to render information back to the user of a mobile device through a Microbrowser.

WML is not an overly complex language, and it benefits from being based on HTML; however, the similarity is only in the structure of the syntax. In Listing 16.1, you can see a simple WML application.

Listing 16.1 **(1601.wml) Simple WML Application**

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
  <card id="main" title="Hello Mobile Device World Example">
     <p>
        Hello Mobile Device User
     </p>
  </card>
</wml>
```

In line 4, you can see the `<wml>` element. This element outlines what is referred to as a *deck*, a collection of cards and forms for a mobile device application. In line 5, we encounter the first `<card>` element. A `<card>` element is used like a page. It contains all the rendering commands for a single screen of data on a mobile device. The WML file can have more than one card nested inside the `<wml>` elements. After the card definition, we use a `<p>` tag to surround any content we need to display. The `<p>` is not optional—it has to be used or the mobile device will not render any of the content for the card. Line 7 contains text to display on the device.

Figure 16.1 shows the screen after the mobile device receives the WML file.

Figure 16.1 WML example.

Enter ASP.NET

So, now you have had a brief look at WAP and WML. It's time to see how this all relates to the Mobile Device SDK for ASP.NET. Table 16.1 outlines some key elements used in WAP/WML development and explains the equivalent elements in ASP.NET:

Table 16.1 **Comparing WAP/WML to ASP.NET**

WAP Element	Mobile Internet Control
`Deck, <wml>`	This is the name of the file sent to a mobile device; the deck can be made up of one or more cards.
`MobilePage`	The `MobilePage` class is similar to the Web Form page class, but it is specifically for the creation of mobile Web Forms.

WAP Element	Mobile Internet Control
`<card>`	The `<card>` is the term given to a chunk of presentation logic for a mobile device page. You can have more than one card to a deck. The first card in the deck file is the first card to be displayed or processed by the mobile device.
`Mobile:Form`	The `Mobile:Form` control encapsulates the functionality of the WML `<card>` element. You can have many forms to a `MobilePage`. The first form is the form that is initially displayed by the mobile device.
`<do>`, `<go>`	When a user interface event occurs, the device performs the associated `<do>` task.
`Mobile:Command`	The `Mobile:Command` control provides a mechanism that enables the user to call an ASP.NET event handler for a task.
`<fieldset>`	The `<fieldset>` element enables you to group multiple text or input items on a card.
`Mobile:Panel`	The `Mobile:Panel` control enables the user to group controls on a form together, so that they can be rendered on one screen (if the mobile device supports that feature).
`<a>`, `<anchor>`	These elements instruct the device to display another card from the current card.
`Mobile:Link`	`Mobile:Link` provides a hyperlink to another file or to another form in the current file.
`<input>`	The `<input>` element provides the user with data entry functionality.
`Mobile:TextBox`	The `Mobile:TextBox` control provides data entry support for text data for the user.
`<select>`	This element renders a list of options from which the user can choose.
`Mobile:List`	The `Mobile:List` control enables the user to select an item from a list of possible values.

Note

You can use `<wml>` elements inside ASP.NET mobile forms without any problems. However, it is better to use ASP.NET controls because the mobile control's toolkit can generate either WML or HTML depending on the capabilities of the device to which it is rendering.

Creating a Mobile Device Application

ASP.NET Mobile controls are defined as elements in exactly the same manner as regular ASP.NET controls. The only thing you have to do with Mobile controls is make sure that at the top of your web page, you register the Mobile Internet Toolkit controls and namespace as shown:

```
01 <%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="vb" %>
02 <%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
03              Assembly="System.Web.Mobile" %>
```

On the first line, the Web Form properties are inherited from the `System.Web.UI.MobileControls.MobilePage` class. This class provides the core functionality of a mobile device web page. Also, the language is set for any script on the web page to use Visual Basic .NET. The second line registers the `TagPrefix` to `mobile` for the namespace `System.Web.Mobile.UI.MobileControls`. This enables access to a Mobile control without having to type in the long namespace first. For more on this, see Chapter 2, "Developing Applications with ASP.NET."

After this has been done, you are ready to start using the Mobile Internet Toolkit controls to build your application.

The *Form* Element

The `Form` element is required for every single mobile device web page without exception. You can have more than one form to a Web Form source file; however, only one form at a time will be rendered on the mobile device. The first form definition in your source file is the initial form that will be displayed on the mobile device. The syntax for the form control follows:

```
01 Mobile:Form runat=server
02             id="id-of-control"
03             StyleReference="StyleReference"
04             OnActivate="OnActivateHandler"
05              OnDeactivate="OnDeactivateHandler">
```

The `id` attribute is used to create a unique identifier for the `Form` in the source file. This is important because you can have many forms in a file, and you will need to reference each form for navigational purposes if nothing else. The `StyleReference` attribute is used to apply a style sheet to any controls inside the `Form` elements. More on this attribute in the "Presentation Controls" section later in this chapter. The `OnActivate` and `OnDeactivate` attributes are used to call a function after these events happen. The `OnActivate` event occurs when the form is first displayed, and the `OnDeactivate` event occurs when the form is replaced by another form.

An example of the `Form` control with an `OnActivate` event handler is defined in Listing 16.2.

Listing 16.2 **An Example of a Simple Mobile Form-Based Application**

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="vb"
      %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
              Assembly="System.Web.Mobile" %>
<script language="vb" runat=server>
  sub One_OnActivate (Source as Object, E as EventArgs)
    ActiveForm = Two
  End Sub
</script>
<Mobile:Form id="One" runat=server OnActivate="One_OnActivate">
  <Mobile:Label runat=server>Form One</Mobile:Label>
</Mobile:Form>
<Mobile:Form id="Two" runat=server>
  <Mobile:Label runat=server>Form Two</Mobile:Label>
</Mobile:Form>
```

Do not be daunted by the amount of code in Listing 16.2. Most of it is straightforward. Lines 1 and 3 set up the Web Form as a mobile Web Form. In line 10, the definition of form `"One"` begins. Notice that the form has an event handler defined for the `OnActivate` event. The handler itself is defined in the script block (lines 5–9); all the handler does is get the mobile device to display another form, in this case `"Two"`, which is defined in line 13.

When this code is run, the screen (shown in Figure 16.2) displays form `"Two"`, not form `"One"`.

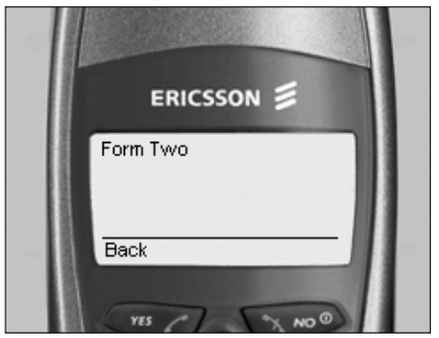

Figure 16.2 `Mobile:Form` example.

Developing Applications for Mobile Devices

The Mobile Internet Toolkit comes with its own collection of web controls, which can be used to create the user interface of a mobile device application. These controls can be separated into three main areas of functionality:

- **Presentation**—Controls used to present content or data to the mobile device display
- **Navigation**—Controls used to navigate from one mobile form to another
- **Data Entry**—Controls used to get input from the user

All the mobile device controls have some default attributes. These are outlined in the following list and apply to all mobile device controls unless otherwise stated:

- **`Id`**—The `Id` attribute is used to create a unique identifier for the control. This is important because you can have many controls on a form, and you may need to reference the label to programmatically change its value.
- **`StyleReference`**—The `StyleReference` attribute is used to apply a stylesheet style to the control.
- **`Runat`**—The `Runat` attribute is required for all mobile device controls, and its value should be set to `Server`, otherwise, the web controls will not be processed and the mobile application will not run.

> **Note**
> All the examples in this section are based on mobile device development for mobile phones and have been tested with the Openwave UP SDK. All the examples work on any WAP 1.1-capable device; however, the rendered results can be different because of the different capabilities of each device.

Presentation Controls

As mentioned previously, the Mobile Internet Toolkit has a collection of controls designed for presenting content on a mobile device. These controls are outlined in the next few sections.

The *Mobile:Label* Control

The `Mobile:Label` control is used to display some static text on the mobile device display area, or it is used as a placeholder for some display text.

The syntax of the `Mobile:Label` control is as follows:

```
01 <mobile:Label runat="server"
02               id="id"
03               StyleReference="StyleRef"
04               Text="Text">
05 Text
06 </mobile:Label>
```

A specific attribute of the `Mobile:Label` control is the `Text` attribute, which is the text to display on the Web Form.

An example of `Mobile:Label` control use is shown in Listing 16.3.

Listing 16.3 ***Mobile:Label* Example**

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="vb"
      %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
              Assembly="System.Web.Mobile" %>
<script language="vb" runat=server>
  sub One_OnActivate (Source as Object, E as EventArgs)
    DynamicLabel.text = "Dynamic text"
  End Sub
</script>
<Mobile:Form id="One" runat=server OnActivate="One_OnActivate">
 <Mobile:Label runat=server Text ="Simple Text" />
 <Mobile:Label runat=server>
 More Text
 </Mobile:Label>
 <Mobile:Label runat=server id="DynamicLabel" text="static text" />
</Mobile:Form>
```

The preceding code demonstrates three different uses of the `Mobile:Label` control. Line 11 uses the `Text` attribute to set the value of the text to display on the device. The second control in line 12 uses both opening and closing elements and enables free text to be typed between them. The final `Label` control in line 15 sets its text value to one value, then has it changed by the form's `OnActivate` event to another value. The script block references the `Label` control by its `id` of `DynamicLabel`, and then set its `Text` attribute to a new value.

Figure 16.3 shows the resulting screenshot of the preceding code on a mobile device.

Figure 16.3 `mobile:Label` example.

The *Mobile:TextView* Control

The `Mobile:TextView` control is used to display multi-line static text on the mobile device display area.

```
01 <mobile:TextView
02     runat="server"
03     id="id"
04     StyleReference="styleReference"
05     Wrapping={NotSet | Wrap | NoWrap}
06     Text="Text">
07 Text
08 </mobile:TextView>
```

The specific attributes of the `Mobile:TextView` control include

- **`Wrapping`**—The `Wrapping` attribute is used to define how the text in the control is wrapped across the device display.
- **`Text`**—The `Text` attribute is the text displayed on the Web Form.

An example of the `Mobile:TextView` control is shown in Listing 16.4.

Listing 16.4 ***Mobile:TextView* Example**

```
 <%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="C#" %>
 <%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
 <mobile:Form runat="server">
   <mobile:TextView runat="server" id="TV" Alignment="Center"
Font-Bold="true">
    This is an example of a mobile:TextView control!
  </mobile:TextView>
 </mobile:Form>
```

The preceding code line demonstrates the use of the `Mobile:TextView` control in lines 4–6. Figure 16.4 shows a resulting screenshot of the code on a mobile device.

Figure 16.4 `mobile:TextView` example.

The *Mobile:Image* Control

The `Mobile:Image` control is used to display images on the mobile device. These images can be from an internal list of symbols on the mobile device, or they can be external graphic files. One additional feature of the `Mobile:Image` control is that you can assign a URL to the image, so it works like a `Mobile:Link` control. (`Mobile:Link` controls are covered later in this chapter in the "Navigation Controls" section.

The syntax of the `Mobile:Image` control is as follows:

```
01 <mobile:Image
02     runat="server"
03     id="id"
04     Alignment={NotSet|Left|Center|Right}
05     StyleReference="styleReference"
06     Wrapping={NotSet|Wrap|NoWrap}
07     AlternateText="AltText"
08     ImageURL="masterImageSource"
09     NavigateURL="targetURL">
10 </mobile:Image>
```

The specific attributes of the `Mobile:Image` control include

- **`Alignment`** The `Alignment` attribute is used to set the physical alignment of the image on the mobile device display area. The `Mobile:Image` can be displayed on the left, right, and center of the screen.

- **`Wrapping`**—The `Wrapping` attribute is used to define how the text in the control is wrapped across the device display.
- **`AlternateText`**—This attribute is used to describe the image. It is rendered to the mobile device as a `Mobile:Label` control, if the mobile device does not support the use of images.
- **`ImageURL`**—The `ImageURL` attribute is a URL of an image file displayed to the user. It can also be an internal symbol name for an internal image on the mobile device. To use a symbol name, you have to prefix the mobile device's internal symbol identifier with `symbol`. An example of the `Mobile:Image` control is in Listing 16.5.
- **`NavigateURL`**—If this attribute is used, the image becomes a link to another Mobile Web Form. The value used with this attribute is the same as the one for the `Mobile:Link` control, which is described in more detail later in the "Navigation Controls" section of this chapter.

In Listing 16.5, we have a more complex sample application.

Listing 16.5 ***Mobile:Image*** **Example**

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="C#" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<mobile:Form runat="server">
<mobile:label runat="server" text="Image Control Example" />
<br>
Open Folder
<mobile:Image runat="server" id="myImage3" ImageURL="symbol:folder2"
            NavigateURL="#form2" />
Closed Folder
<mobile:Image runat="server" id="myImage2" ImageURL="symbol:folder1"
            NavigateURL="#form3" />
</mobile:Form>

<mobile:Form id="form2" runat="server">
<mobile:TextView runat="server"  font-bold="True"
                  text="You Selected the open Folder" />
</mobile:Form>
<mobile:Form id="form3" runat="server">
<mobile:TextView runat="server" font-bold="True"
                  text="You Selected the Closed Folder" />
</mobile:Form>
```

In this application, I have added a `Mobile:Label` control (line 4) to be used as the title of the application. Then two image controls follow (lines 7–8),

which are using the mobile device's internal symbol library and have been assigned an internal form URL to jump to once they have been selected. The output of these lines is shown in Figures 16.5 and 16.6.

Figure 16.5 `Mobile:Image` example, first page.

Figure 16.6 `Mobile:Image` example, after selection.

The *Mobile:Panel* Control

`Mobile:Panel` controls are used to group controls together and also to apply styles to a group of controls. Listing 16.6 uses a panel control to apply a style to nested controls in two different panels.

The syntax of a `Mobile:Panel` control is as follows:

```
01 <mobile:Panel
02     runat="server"
03     id="id"
04     Alignment={NotSet¦Left¦Center¦Right}
05     StyleReference="styleReference" >
06 … Controls inside Panel Control go here
07 </mobile:Panel>
```

The specific attributes of the `Mobile:Panel` control include the `alignment` attribute, which is used to set the physical alignment of the controls inside the panel on the mobile device display area. The `Mobile:Image` can be displayed on the left, right, or center of the screen.

Listing 16.6 shows a simple example of using the `Mobile:Panel` control to apply group formatting.

Listing 16.6 ***Mobile:Panel* Example**

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="c#" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile"  %>
<mobile:Form runat="server">
<mobile:Panel runat="server" Font-Italic="true" Alignment="left">
  <mobile:Label runat="server">First Panel</mobile:Label>
  <mobile:Label runat="server">is up here</mobile:Label>
</mobile:Panel>
<mobile:Panel runat="server" Font-Bold="true" Alignment="right">
 <mobile:Label runat="server">Second Panel here</mobile:Label>
</mobile:Panel>
</mobile:Form>
```

The first `Mobile:Panel` control (lines 4–7) sets the font to italic and also fixes the alignment to the left side. For all controls in the panel, the second `Mobile:Panel` control (lines 8–10) has only one control, which is right-aligned and set to bold. Figure 16.7 shows the output.

Figure 16.7 `Mobile:Panel` example.

The *Mobile:StyleSheet* Control

This control is used to create user-defined style sheets to apply to mobile device controls.

The syntax of the `Mobile:StyleSheet` control is as follows:

```
01 <mobile:Stylesheet
02     runat="server"
03     id="id"
04     Font-Name="fontName"
05     Font-Size={NotSet|Normal|Small|Large}
06     Font-Bold={NotSet|False|True}
07     Font-Italic="{NotSet|False|True}
08     ForeColor="foregroundColor"
09     BackColor="backgroundColor"
10     Alignment={NotSet|Left|Center¦Right}
11     StyleReference="styleReference"
12     Wrapping={NotSet|Wrap|NoWrap}
13     ReferencePath="externalReferencePath" >
14 </mobile:Stylesheet>
```

The specific attributes of the `Mobile:StyleSheet` control include the following:

- **`Font-Name`**—This attribute enables you to select the name of the font you want to use. For most mobile phones, this attribute does nothing because it generally only supports one font.
- **`Font-Size`**—This attribute enables you to select the size of the font used on the mobile device. Again, this has very little use on mobile phones because the display area is so small.
- **`Font-Bold`**, **`Font-Italic`**—These two attributes act like switches. They should be set to either `true` or `false`.
- **`ForeColor`**, **`BackColor`**—These are used to set the foreground and background colors for text display on a mobile device, for mobile phone development, they are not very useful but because most mobile devices are monochrome in nature.
- **`StyleReference`**—The `StyleReference` control can be used to inherit the style settings from another stylesheet control.
- **`ReferencePath`**—This attribute holds a relative path to a user control (.ascx file), which contains a set of style elements. These style controls can then be used in the current mobile application file.

In Listing 16.7, we declare a style sheet at the beginning of the code (lines 3–6), which has two styles defined—`Style1` and `Style2`, respectively. These styles only apply font-bold and font-italic style properties, but the styles can

use any combination of styles shown in the control's syntax. After the styles have been defined, simply apply them to any control we want by using the `StyleReference` attribute of the control. Listing 16.7 uses `Mobile:Label` controls.

Listing 16.7 ***Mobile:StyleSheet* Example**

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="C#" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<mobile:StyleSheet runat="server">
   <Style Name="Style1" font-bold="true"/>
   <Style Name="Style2" font-italic="true"/>
</mobile:StyleSheet>
<mobile:Form runat="server">
<mobile:Label runat="server" StyleReference="Style1">
               This is Style 1</mobile:Label>
<mobile:Label runat="server"  StyleReference="STyle2">
               This is Style 2</mobile:Label>
</mobile:Form>
```

The output is shown in Figure 16.8.

Figure 16.8 `Mobile:StyleSheet` example.

Navigation Controls

There are two types of navigation controls in a mobile web application:

- **`Internal navigation`**—This is used to change the currently displayed form to another form in the same .aspx file.
- **`External navigation`**—This is how you navigate to the first form in another .aspx file.

To use internal navigation, you must prefix the form that you want to navigate with a "#" symbol, and to navigate to an external form, just supply the filename:

- **`#Form1`**—Navigates to the form with the ID `form1`.
- **`File1.aspx`**—Navigates to the first available form in the File1.aspx file.

The *Mobile:Link* Control

The `Mobile:Link` control is used to display a text label that operates as a hyperlink to another form in the same file or external document.

The syntax of the `Mobile:Link` control is as follows:

```
01 <mobile:Link
02     runat="server"
03     id="id"
04     Text="Text"
05     NavigateURL="relativeLink"
06     SoftkeyLabel="softkeyLabel">
07 </mobile:Link>
```

The specific attributes and properties of the `Mobile:Link` control include

- **`Text`**—Sets the text to display the link on the mobile device.
- **`NavigateURL`**—Holds the URL of the form that you want to render. If the value of the NavigateURL property begins with a (#), the remainder of the value is assumed to be the identifier of a form on the current `Mobile:Page` control. Otherwise, the value of the NavigateURL property is treated as a standard URL.
- **`SoftKey`**—Holds the text that is displayed above the softkey of a softkey-capable mobile device.

Listing 16.8 shows a single link control that, when selected, will navigate to the 123 Jump news and finance WAP portal. See Figures 16.9 and 16.10 to see how this appears on your mobile device.

Listing 16.8 ***Mobile:Link* Example**

```
 <%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="C#" %>
 <%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
 <mobile:Form runat="server">
 <mobile:Link runat="server"
    NavigateURL="http://www.123jump.com">123 Jump WAP Portal</mobile:Link>
 </mobile:Form>
```

Figure 16.9 `Mobile:Link` example, link page.

Figure 16.10 `Mobile:Link` example, `www.123Jump.com`.

The *Mobile:Command* Control

The `Mobile:Command` control is used to display a text label that operates as a hyperlink to another form in the same file or an external document.

The syntax of the `Mobile:Command` control is as follows:

```
<mobile:Command
    runat="server"
    id="id"
    Text="text"
    CommandArgument="commandArgument"
    CommandName="commandName"
    OnClick="clickEventHandler"
    OnItemCommand="commandEventHandler"
    SoftkeyLabel="softkeyLabel">
</mobile:Command>
```

The specific attributes and properties of the `Mobile:Command` control include

- **`Text`**—This attribute sets the text to display as the command on a mobile device.
- **`OnClick`**—Holds the name of a sub, which is to be called when the user selects a `Mobile:Command` control.
- **`SoftKey`**—Holds the text that is displayed above the softkey of a softkey-capable mobile device.

Listing 16.9 shows an example of the `Mobile:Command` control, which basically requests both the user's first and last names (lines 11 and 13), and then when the command button is selected (line 14), the procedure `myCmd_OnClick` (lines 03–07) is called, and a second form is subsequently displayed with values entered by the user (lines 17–19). See Figures 16.11, 16.12, and 16.13.

Listing 16.9 ***Mobile:Command*** **Example**

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="vb" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<script language="vb" runat="server">
Protected Sub myCmd_OnClick(sender As Object, e As EventArgs)
   myLabel.Text = "Hello: " + FNEdit.Text + " " + LNEdit.Text
   ActiveForm = form2
end sub
</script>
<mobile:Form runat="server">
    <mobile:Label runat="server" font-bold="True">Enter First Name</mobile:Label>
   <mobile:TextBox runat="server" id="FNEdit" />
   <mobile:Label runat="server" font-bold="True">Enter Last Name</mobile:Label>
   <mobile:TextBox runat="server" id="LNEdit" />
   <mobile:Command runat="server" id="myCmd" OnClick="myCmd_OnClick">
       OK
   </mobile:Command>
</mobile:Form>
<mobile:Form runat="server" id="form2">
   <mobile:Label runat="server" id="myLabel" />
</mobile:Form>
```

The *Mobile:Image* Control

This control has already been covered earlier in this chapter, but you can use the `NavigateURL` attribute to assign a link to which a page can navigate.

The *Mobile:List* Control

You can also use the `Mobile:List` control to navigate to other pages. This is covered in the "*Data Entry* Controls" section of this chapter.

Figure 16.11 `Mobile:Command` example, first name.

Figure 16.12 `Mobile:Command` example, last name.

Figure 16.13 `Mobile:Command` example, hello message.

The Code-Based Alternative

You also can navigate to another form within the current mobile page by setting the `ActiveForm` property of the `mobilePage` class. This property takes the name of a form to navigate to as its value. After the property is assigned, the new form is displayed on the device screen. An example call is shown by the following:

```
ActiveForm = Form1
```

The preceding code tells the mobile device to display a form with the ID of `Form1`.

Data Entry Controls

The Mobile Internet Toolkit supports data entry through two data entry controls:

- `Mobile:TextBox`
- `Mobile:List`

Although this seems to be a very limited set of controls, you must not forget that the browser display is limited in display area size, and mobile devices also generally have limited processing power and memory. However, with some ingenuity, you can create an intuitive user interface for data entry.

The *Mobile:TextBox* Control

The `Mobile:TextBox` control is used to get information from the user of a mobile device. It supports only two types of data entry: general text and password formats. The control enables you to specify formatting rules for the text entered, and you can also limit the amount of text entered in the control as well as the physical size of the control in characters.

The syntax of the `Mobile:TextBox` control is as follows:

```
01 <mobile:TextBox
02     runat="server"
03     id="id"
04     MaxLength="maxLength"
05     Numeric="{true, false}"
06     Password="{true, false}"
07     Size="textBoxLength"
08     Text="Text">
09 </mobile:TextBox>
```

The specific attributes of the `Mobile:Textbox` control include

- **`Text`**—This attribute sets the text to display as the mobile device.
- **`MaxLength`**—This attribute is used to define how much text can be entered into the `Mobile:TextBox` control. A value of **`0`** means there is no length restriction for the textbox.

- **`Numeric`**—This attribute tells the textbox to only accept numeric input. It can be set to either `true` or `false`.
- **`Password`**—This attribute tells the textbox to `mask all data entry` as though a password were being entered. (The characters typed are echoed back to the display as `*` characters.)

Listing 16.10 demonstrates three different uses of the `Mobile:TextBox` control. These are normal data entry, password data entry, and numeric-only data entry. Figures 16.14, 16.15, 16.16, and 16.17 show the output for this code.

Listing 16.10 ***Mobile:TextBox* Example**

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="vb" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<script language="vb" runat="server">

protected sub Button_OnClick(o as Object, e as EventArgs)
   ActiveForm=form2
   lblName.Text = "Name: " + txtName.Text
   lblPassword.Text = "Password: " + txtPassword.Text
   lblBalance.Text = "Bank Balance: " + txtBalance.Text
end sub
</script>
<mobile:Form runat="server">
   <mobile:Label runat="server" font-Bold="True">
Enter your name</mobile:Label>
   <mobile:TextBox runat="server" id="txtName" />

   <mobile:Label runat="server" font-Bold="True">Enter your password</mobile:Label>
   <mobile:TextBox runat="server" id="txtPassword" password="true"/>

   <mobile:Label runat="server" font-Bold="True">Enter your Bank Balance</mobile:Label>
   <mobile:TextBox runat="server" id="txtBalance" numeric="true" />

   <mobile:Command runat="server" id="Button" OnClick="Button_OnClick">
       OK
   </mobile:Command>
</mobile:Form>
<mobile:Form id="form2" runat="server">
   <mobile:Label runat="Server" font-bold="True" Text="You entered" />
   <mobile:Label runat="server" id="lblName" />
   <mobile:Label runat="server" id="lblPassword" />
   <mobile:Label runat="server" id="lblBalance" />
</mobile:Form>
```

Figure 16.14 `Mobile:TextBox` example, name.

Figure 16.15 `Mobile:TextBox` example, password.

Figure 16.16 `Mobile:TextBox` example, bank balance.

Figure 16.17 `Mobile:TextBox` example, display all entered text.

The *Mobile:List* Control

The `Mobile:List` control can be used to render a list of items to the mobile device user. These are either static or loaded into the control from a data source.

The syntax of the `Mobile:List` control is as follows:

```
01 <mobile:List
02     runat="server"
03     id="id"
04     DataTextField="dataTextField"
05     DataValueField="dataValueField"
06     OnItemCommand="onItemCommandHandler"
07 …Mobile item controls that make up the list control
08 </mobile:List>
```

The specific attributes and property of the `Mobile:List` control include:

- **`DataTextField`**—Specifies which property of a data bound item to use when determining an item's text property.
- **`DataValueField`**—Specifies which property of a data bound item to use when determining an item's value property.
- **`OnItemCommand`**—Holds the name of the event handler to be called when an individual list item generates an event.
- **`Datasource`**—Holds the data source to which the control is bound.

Listing 16.11 uses a `Mobile:List` control to display a list of books. After an item is selected, the selected book title and the author who wrote it are displayed.

Listing 16.11 ***Mobile:List*** **Example**

```
<%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="vb" %>
<%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<script runat="server" language="vb">
protected sub List_EventHandler(o as Object, e as ListCommandEventArgs)
   lblBook.Text = e.ListItem.Text
   lblAuthor.Text = e.ListItem.Value
   ActiveForm = form2

end sub
</script>
<mobile:Form id="myList" runat="server">
<mobile:Label runat="server" id="label1" text="Select Book"/>
<mobile:List runat="server" id="ListProduce" OnItemCommand="List_EventHandler" >
<item Text="Inside ASP.NET" Value="Scott Worley" />
<item Text="Inside XSLT" Value="Steven Holzner" />
<item Text="Inside XML" Value="Steven Holzner" />
</mobile:List>
</mobile:Form>
<mobile:Form id="form2" runat = "server">
    <mobile:Label runat="server" font-bold="True" Text="Book Selected"/>
    <mobile:Label runat="server" id="lblBook"/>
    <mobile:Label runat="server" font-bold="True" Text="Author"/>
   <mobile:Label runat="server" id="lblAuthor"/>
</mobile:Form>
```

On line 13, I have created a simple `Mobile:List` control and assigned the event handler sub `List_EventHandler` to process any selection made from the list. On lines 14–16, I have assigned items to the list. When the form is run, the list is displayed, and when an item has been selected, the eventhandler will call the `List_EventHandler` sub at the beginning of the code listing. This sub reads information from the `Mobile:List` control and assigns its value to a `Mobile:Label` control on the form `form2`. Figures 16.18 and 16.19 show the output to this code.

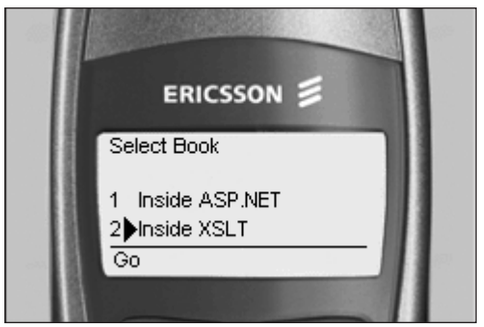

Figure 16.18 `Mobile:List` example, book selection list.

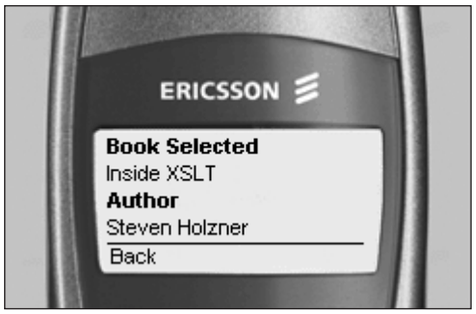

Figure 16.19 `Mobile:List` example, display selection details.

You also can bind the `Mobile:List` control to a data source. Listing 16.12 demonstrates this technique.

Listing 16.12 **Databound *Mobile:List* Example**

```
 <%@ Page Inherits="System.Web.UI.MobileControls.MobilePage" Language="vb" %>
 <%@ Register TagPrefix="mobile" Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
 <%@ import namespace="System.Data.SqlClient" %>
 <script runat="server" language="VB">

 Protected Sub Page_Load(sender As Object, e As EventArgs)
  If (Not IsPostBack)
    Dim connString as String
    Dim sqlString as String
    Dim myConn as SQLConnection
    Dim myCmd as SQLCommand
    Dim myReader as SQLDataReader

    connString = "Initial Catalog=Pubs;Data Source=p450;uid=sa;pwd=;" 15    sqlString =
"SELECT fname+' '+lname as name, hire_date FROM employee"
    myConn = new SqlConnection(connString)
    myCmd = new SqlCommand(sqlString, myConn)
    myConn.Open()

    myReader = myCmd.ExecuteReader()
    MenuList.DataSource = myReader
    MenuList.DataBind()
    myReader.Close()

    End If
 End Sub
 Protected Sub Menu_OnItemCommand(sender As Object, e As ListCommandEventArgs)
    lblEmployee.Text = e.ListItem.Text
```

```
   lblHireDate.Text = Ctype(e.ListItem.Value, String)
   ActiveForm = form2
End Sub
</script>
<mobile:Form runat="server">
   <mobile:Label runat="server" StyleReference="title">Select Employee</mobile:Label>
   <mobile:List runat="server" id="MenuList" OnItemCommand="Menu_OnItemCommand"
       DataTextField="name" DataValueField="hire_date" />
</mobile:Form>
<mobile:Form id="form2" runat="server">
   <mobile:Label runat="server" font-bold="True" Text="Employee:" />
   <mobile:Label runat="server" id="lblEmployee" />
   <mobile:Label runat="server" font-bold="True" Text="Date Hired:" />
   <mobile:Label runat="server" id="lblHireDate" />
</mobile:Form>
```

The preceding listing uses a `Page_Load` event handler to connect to a SQL Server database—in this case, the pubs database, which is supplied with `SQLServer`. A `SQLDataReader` is used to get data from the Employees table, and then subsequently is bound to the `Mobile:List` control defined on line 47. The data binding process used with mobile controls is the same as that used by regular ASP.NET forms. One thing to note are lines 47 and 48.

Of special interest here are the last two attributes, `DataTextField` and `DataValueField`. These attributes are the column names from the SQL query with which the `Mobile:List` control is bound. For the binding to work, you must have these two attributes. See Figures 16.20 and 16.21.

Figure 16.20 Databound `Mobile:List` example, employee selection.

Figure 16.21 Databound `Mobile:List` example, display selection details.

Data binding is covered in a lot more detail in Chapter 6, "Using ADO.NET in ASP.NET Applications."

Summary

As you have seen, the Mobile Internet Toolkit enables you to create very functional mobile applications very quickly and without the need to understand the inner workings of the devices you develop. However, in this chapter, I have only scratched the surface of what is possible with the toolkit. You can get further information from the online documentation.