



CHAPTER 5

Getting information

5.1 Introduction	81	5.4 Restricting data entry	89
5.2 About menus	82	5.5 Images	91
5.3 Using input fields	87	5.6 Summary	94

5.1 INTRODUCTION

Chapter 4 described navigation through an application—how to enable a user to interact with an application by moving among the cards that make up a deck. Chapter 5 expands the discussion of user interaction by describing how to develop ways to enter data. In chapter 4 we also discussed how to use menus to allow a user to navigate to different cards. That part of menus will not be covered again here. In chapter 5 we will concentrate on using menus to select data.

After completing this chapter, you should be able to build applications that interact with the user, enabling them to make choices and enter data. You will also learn how to work with images in a WAP application. Carefully used, providing images to supplement or replace text on the phone's display can be a powerful means to improve a user's experience with your application.

5.2 ABOUT MENUS

Because entering data on a phone is difficult, presenting a menu from which users can make selections is one of the most important means by which to interact with users. In some cases, a menu might be creatively used to replace data entry altogether. For example, to avoid forcing the user to enter the name of a state by hand an alternative is to offer a series of menus that move from general to specific—the user would choose an area (northwest, southeast, etc.) and then be presented with a choice of states within the selected area.

5.2.1 Overview of the WML menu elements

Creating a typical menu is a two-step process. First, define the boundaries of the menu with the `<select>` element—all menu items will be contained within `<select>...</select>`. Second, the individual menu selections are defined using `<option>` elements nested within the `<select>` element.

WML includes menu-specific elements (table 5.1).

Table 5.1 WML menu elements

WML Element	Description
<code><select></code>	All menus are contained within the <code><select></code> element, which delimits the menu and whose attributes define the data for the menu. For example, you can assign a value to a variable based on the user's selection within a menu. The name of the variable and its default value are defined within <code><select></code> .
<code><option></code>	Defines the specific choices within a menu. <code><option></code> always appears nested within the <code><select></code> element.
<code><optgroup></code>	Nests menus such that selecting a menu item presents the user with an additional menu. An example of this concept: A user would select a large geographic area from a menu, and would then be presented with another menu displaying states within the selected area.

5.2.2 Using the `<select>` element

To prompt a user to choose one or more items from a specified list, use the `<select>` element. Its attributes are shown in table 5.2.

Table 5.2 Attributes for a `<select>` element

Attribute	Description
Title	Specifies a label for the menu. This attribute is used differently on different devices.
Multiple	Specifies whether the user can choose multiple items. The default is <code>false</code> (meaning that choosing multiple items is not allowed).
Name	Specifies the name of the variable in which the value of the user's choice is stored. In the case of multiple selection lists, the value is stored as a semicolon-separated list.
Value	Specifies the default value to store in the variable specified in the <code>name</code> attribute. In the case of multiple selection lists, the value is a semicolon-separated list.

Table 5.2 Attributes for a <select> element (continued)

Attribute	Description
iname	Specifies the name of the variable that stores the index of the user's choice. The indexes start at 1 and in the case of multiple-selection lists, the values are stored as a semicolon-separated list.
lvalue	Specifies a string containing the default value for an iname variable (in other words, the default index values). Again, in the case of multiple-selection lists, the value is a semicolon-separated list.

In the next section you will see examples of how each of these attributes can be used.

5.2.3 Using the <option> element

Contained within the content of the <select> element, which defines the entire menu, are the <option> elements, which define specific choices. The <option> element contains an attribute, `value`, which enables you to specify a value to assign to the variable that was defined in the <select> element's name attribute.

Now let's combine what we know about <select> and <option> elements to create useful menus. In this first example we use only the name attribute of the <select> element and the value attribute of the <option> element to create a single-choice list. The user will choose Film, Director, or Actor and the value of each choice will be put in the variable choice. This value can now be used elsewhere in the application. Variables will be discussed in detail in chapter 6. The point here is to learn how to implement menus to get a user's input. Here is the code:

```
<wml>
<card>
<do type="accept">
  <go href="#display"/>
</do>
<p>
  Search for:
  <select name="choice">
    <option value="film">Film</option>
    <option value="director">Director</option>
    <option value="actor">Actor</option>
  </select>
</p>
</card>

<card id="display">
<p>
  You chose $(choice).
</p>
</card>
</wml>
```

Defines the variable as "choice"

Sets the variable "choice" equal to "film"

Displays the value that the user chose

Figure 5.1 displays the result.

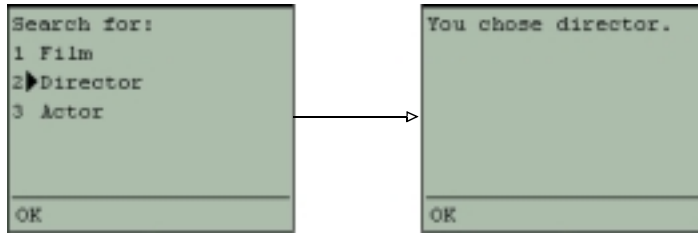


Figure 5.1 Choosing a menu option

To create a multiple-selection list we use the name and multiple attributes of the `<select>` element and the value attribute of the `<option>` element. In this case, the user can choose any or all of the selections (Film, Director, and Actor). The variable, in this case `choice`, will hold the values of the selected items in a semicolon-separated list, which can be used elsewhere in the application. Again, don't worry about the use of the variable; the point is to understand the use of a multiple-selection list. Here is the code:

```

<wml>
<card>
<do type="accept">
  <go href="#display"/>
</do>
<p>
  Search for:
  <select name="choice" multiple="true">
    <option value="film">Film</option>
    <option value="director">Director</option>
    <option value="actor">Actor</option>
  </select>
</p>
</card>

<card id="display">
<p>
  You chose $(choice).
</p>
</card>
</wml>

```

Defines the variable as "choice" and multiple is set to "true"

Sets the variable "choice" equal to "film;director" if both of these are selected by the user

Displays the value that the user chose

Figure 5.2 shows multiple selections.

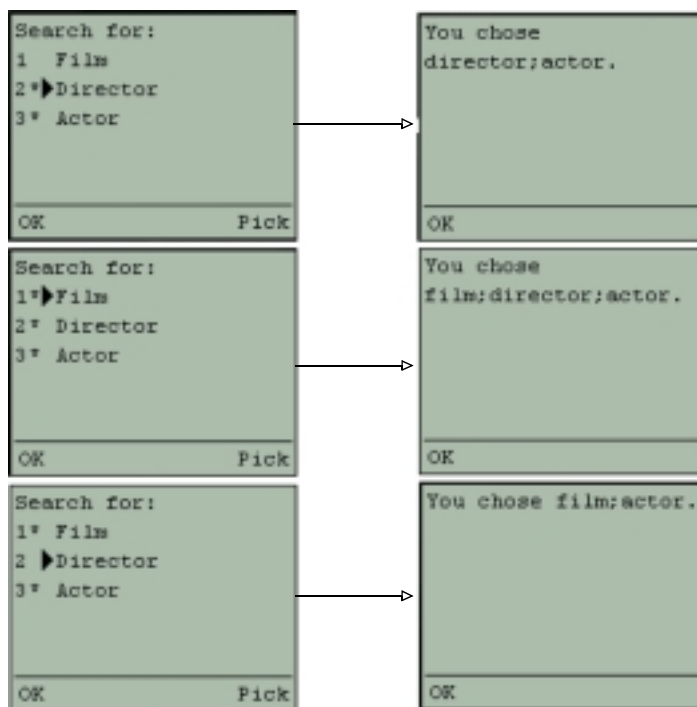


Figure 5.2
Multiple selections

Our final example demonstrates how to use the `iname` attribute of the `<select>` element. In this example, instead of displaying the actual choice that the user makes, we are going to display the indexes that they chose. This information, held in the variable defined in the `iname` attribute of the `<select>` element, can then be used elsewhere in your application. Here is the code:

```

<wml>
<card>
<do type="accept">
<go href="#display"/>
</do>
<p>
  Search for:
  <select iname="choice" multiple="true">
    <option value="film">Film</option>
    <option value="director">Director</option>
    <option value="actor">Actor</option>
  </select>
</p>
</card>

<card id="display">
<p>
  You chose $(choice).

```

Defines the variable as "choice" and multiple is set to "true"
 Sets the variable "choice" equal to "1"
 Displays the index value that the user chose

```

</p>
</card>
</wml>

```

Figure 5.3 shows the result of using the `iname` attribute in the `<select>` element.

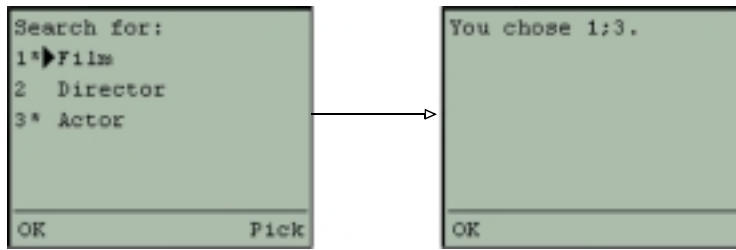


Figure 5.3 Using `iname` in the `<select>` element

5.2.4 Using the `<optgroup>` element

The `<optgroup>` element enables you to create nested option lists such that a menu option is associated with another menu. That is, the user is presented with a menu, and upon choosing an item from the menu is presented with another (subordinate) menu. In the following example, a user first chooses a set of states (Some states or Other states) and is then presented with a list of states.

Nesting menus can be a good technique for avoiding data entry. As the next code snippet and figure 5.4 show, nested menus allow the user to choose their home state from a series of menus (instead of being prompted with a form inviting the user to enter a state name using the phone's keypad).

```

<wml>
<card>
<do type="accept">
  <go href="#display" />
</do>
<p>
  Search for:
  <select name="choice">
    <optgroup title="Some states">
      <option value="ME">Maine</option>
      <option value="NH">New Hampshire</option>
      <option value="VT">Vermont</option>
    </optgroup>
    <optgroup title="Other states">
      <option value="CT">Connecticut</option>
      <option value="MA">Massachusetts</option>
      <option value="RI">Rhode Island</option>
    </optgroup>
  </select>
</p>

```

Defines the first menu

Displays the associated menu— its choices are specified by each subsequent `<option>` element

```

</card>
<card id="display">
<p>
    You chose $(choice).
</p>
</card>
</wml>

```

Here's how the nested menus appear on the phone.

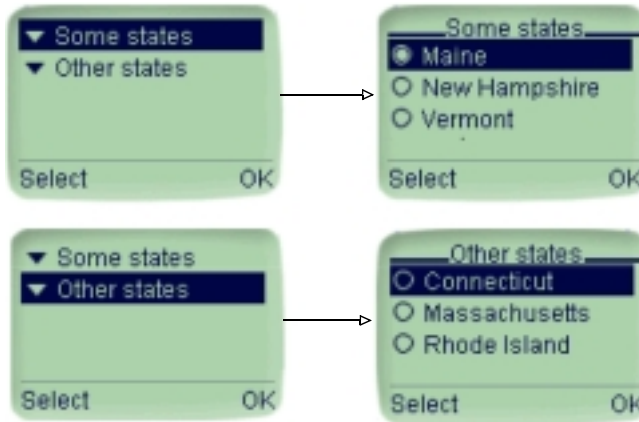


Figure 5.4
Using `<optgroup>`

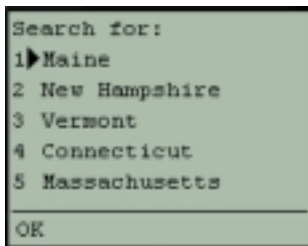


Figure 5.5 `<optgroup>` on a Phone.com browser

NOTE

Some devices (specifically those with the Phone.com browser) will not render `<optgroup>` correctly—that is, as different menus presented on separate cards. Instead, they will display all options simultaneously, ignoring the `<optgroup>` element (always test your code on multiple browsers). Figure 5.5 shows how the `<optgroup>` code appears on a Phone.com browser.

5.3 USING INPUT FIELDS

Phone keypads are designed to enable a user to easily enter someone's phone number, not text. Entering text on a phone is not only difficult, but can be dangerous if the user is attempting to enter text while driving. For example, to enter the letter *s* on a phone, you must press the 7 key four times. Now try entering Mississippi on your phone—hard to do! Therefore, limit data entry in your application whenever possible. If you are designing an application that requires text entry, carefully consider whether there is any possible means by which to extract the same information using

some other method (with menus, perhaps, or by giving that user an option to set frequently used variables on a PC). In our “Mississippi” example, we could use MS.

5.3.1 Using the `<input>` element

To enable users to enter data directly into the phone, use the `<input>` element. For all the attributes that the `<input>` element supports, see appendix A. For our purposes, the `name` attribute is all that we need. It specifies the name of the variable that is going to be filled with the data that the user enters. This functionality is similar to data entry on HTML forms. Although forcing a user to enter text into a phone is best avoided, there are circumstances that require it. If you have a secured application, for example, you might require a user to enter a user name and password.

The following example demonstrates using the `<input>` element to prompt the user for a search string:

```
<wml>
<card id="srchfor">
<do type="accept">
  <go href="#display"/>
</do>
<p>
  Search for: <br/>
  <input name="srchfor"/>
</p>
</card>

<card id="display">
<p>
  Search for:
  $(srchfor)
</p>
</card>
</wml>
```

← Prompts user to enter a search string

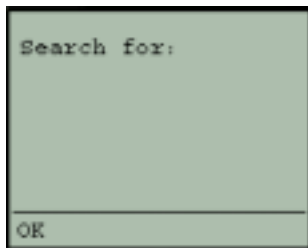


Figure 5.6 Input card on a phone’s browser

Figure 5.6 shows the input card. Note that the input field typically is not delimited by a box the way it is on a PC. Many phones or emulators display the input field differently.

There are two important aspects to data entry that a developer should understand: how to enable the user to enter data and what to do with the data once it is entered. Please consult chapter 6 for more information on the latter.

By default, when a phone encounters the `<input>` element, it requires the user to enter data; you can, however, override this behavior by setting the `emptyok` attribute to true within the `<input>` element. Setting this attribute enables the user to navigate beyond the input card without entering any data (without the attribute, nothing will

happen if the user presses OK without entering data). For example, let's modify the search form in the earlier example with this attribute:

```
<card id="srchfor">
<do type="accept">
  <go href="#display"/>
</do>
<p>
  Search for: <br/>
  <input name="srchfor" emptyok="true"/>
</p>
</card>
```

The `emptyok = true` attribute permits the user to continue without entering any data

In this case, the user can submit a request to the database without entering a search term. The default setting is `emptyok=false`, forcing users to enter a search term before proceeding. Setting the `emptyok` attribute to `true` is valuable in some applications—for example, a search form submitted without data might return a list of all entries within the database.

5.4 RESTRICTING DATA ENTRY

Because of the limited space available on the phone's display, it can be difficult to effectively describe to a user the type of data the application expects, resulting in data entry that makes no sense to the application. For example, should a birth date be entered as `March 07, 1968`, `030768`, `3/7/68`, or another format altogether?

If your application depends on a specific data format, the `<input>` element provides attributes that restrict users from entering invalid data types.

5.4.1 Formatting input fields

In addition to serving as a prompt for users to enter data, input fields can also require the user to enter data using a specific format. That is, the input field specifies whether the user can enter numbers, letters, or any combination thereof. The `format` attribute of the `<input>` element enables you to specify the type, case, and number of characters the user can enter.

To format an input field, specify a combination of the special characters shown in table 5.3, using the `format` attribute.

Table 5.3 Formatting input fields

Character	Description	Example
A	Allows any uppercase alphabetic character (no numbers)	Format=AAAA Allows four uppercase characters
a	Allows any lowercase alphabetic character (no numbers)	Format=aa Allows two lowercase characters
N	Allows any numeric character (no symbols)	Format=NN Allows two numbers

Table 5.3 Formatting input fields (continued)

Character	Description	Example
X	Allows any numeric, symbolic, or uppercase alphabetic character	Format=XX Allows two of any character except lowercase
x	Allows any numeric, symbolic, or lowercase alphabetic character	Format=x Allows one of any character except uppercase
M or m	Allows any alphabetic character (of any case), and any numeric or symbolic character	Format=Mm Allows two of any character

These formats can be combined to create an infinite number of arrangements as well. For example, entering the attribute `FORMAT=NNAA` specifies that the user must enter exactly two numbers followed by exactly two uppercase alphabetic characters. When you indicate specific entry, the Accept button will not work (nor will the Accept label be displayed) until the appropriate data quantity and type have been entered.

To enable a user to enter a particular number of characters of a particular type, use the format `n`type, where *n* is the maximum number of characters allowed and type is the character type allowed (*A*, *a*, *N*, *X*, *x*, or *M*). To indicate an unlimited number of entries, use an asterisk (*) in place of *n*. For example, `format=N5M` enables the user to enter one digit followed by up to five alphanumeric characters, while `format=N*M` enables the user to enter one digit followed by *any* number of alphanumeric characters.

NOTE You can use an asterisk or a number only with the **last** character of a format string. For example, the format string `4AN` *does not* work. To allow the user to enter four uppercase alphabetic characters followed by a numeric character, you must use the string `format=AAAAN`.

5.4.2 Preformatting entry data

It is often helpful to place automatic characters that the user cannot edit into an input field. To preformat data, insert a backslash (\) within the `format` attribute followed by the character that should be automatically generated. When the user enters data, the phone automatically inserts those characters at the positions specified by the backslash.

For example, suppose you use `format=\(NNN\)`. This instructs the phone to automatically insert a left parenthesis before the user has entered anything and a right parenthesis after the user enters three digits (as with an area code).

Similarly, if the user is entering a Social Security number, you might want the phone to automatically place hyphens between the number groupings. The following example shows how you can define specific formatting for a Social Security number using the `format` attribute of the `<input>` element:

```
<wml>
<card id="SSN">
<do type="accept">
```

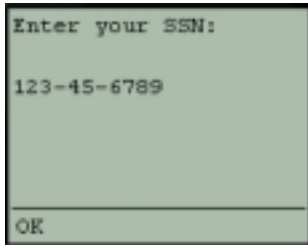
```

    <go href="#display" />
</do>
<p>
  Enter your SSN: <br/>
  <input name="SSN" format="NNN\ -NN\ -NNNN" />
</p>
</card>

<card id="display">
<p>
  Your SSN: <br/>
  $(SSN)
</p>
</card>
</wml>

```

Creates a variable into which the value of the input field will be stored. The format attribute automatically inserts a dash (-) in the standard positions for an SSN



The display will look like figure 5.7 when data entry is complete. The hyphens appear automatically as the user enters data.

Figure 5.7 Preformatting entry data

5.4.3 Ordering input fields

If you create an input form by specifying multiple `<input>` or `<select>` statements in a single card, it is likely that many users will not be able to see all fields simultaneously because the phone's display area is too small to contain them all. By setting the `order` attribute at the card level, you can specify if the phone should display the fields in separate cards or within the same card (through which a user can scroll).

- `<card ordered=true>` (*the default setting*)—Generally best for short forms containing (mostly) required fields—this choice creates a linear sequence of screens through which users must navigate in a fixed order.
- `<card ordered=false>`—Better for longer forms containing fields that are (mostly) optional or have no sequential order—this attribute value creates a menu that lets users navigate to different fields in any order.

5.5 IMAGES

Images are an important part of the PC-based Web, and can be used to great effect on the phone as well. For example, images can be used to supplement or replace text, or to enhance the branding of an application.

Before discussing how to do this, some cautions are in order: Use images thoughtfully—the limitations of the phone (both the small display size and limited memory) require images that are very compact and fit within the context of the application. When using images, remember that an image that works well on a particular model might render the application unusable on a phone with a more modest display. With images, being conservative is the best strategy.

5.5.1 Using the element

Images are placed on the display with the element, which must be nested within a <p> element, similar to regular text. The `src` attribute within the element defines the location of the image (the URL where the image can be found).

The following example shows how an image might be used in the opening card of a movie directory site:

```
<wml>
<card id="welcome">
<p>
  <center>
    Movie lookup!<br/>
    
  </center>
</p>
</card>
</wml>
```

← Displays the image at the indicated URL on the phone

The resulting image is shown in figure 5.8.

You can make the image a link by nesting it within an <a> element. In the following example, the movie lookup card has been altered such that the image of the movie reel is a link:



```
<wml>
<card id="welcome">
<p>
  <center>
    Movie lookup!<br/>
    <a href="#search by">
      <img src=http://www.domain.com/movie/splash.bmp
        alt="Movie Logo"/>
    </a>
  </center>
</p>
</card>
</wml>
```

← Nesting the image within the <a> element points the image at the specified URL

Figure 5.8
Displaying an image

5.5.2 The WBMP image format

Because the common image formats used on the Web (GIF and JPEG in particular) are ill-suited for transmission to phones, the WAP Forum has defined an image

format, WBMP (or wireless bitmap) specific for wireless devices. WBMP is a limited format that contains no compression and does not support color.

Tools to convert images to the WBMP format are widely available as downloads. Because WBMP tools are constantly being introduced and updated, any attempt to list them here would be immediately obsolete. Run a search on WBMP tools in your favorite search engine, and you'll find many available.

5.5.3 Using the ALT attribute

Although all WAP-compatible gateways must recognize WBMP as a legitimate MIME type, support for a particular graphic file format (if graphics are supported at all) is not standardized across phone models. Because of this, get into the habit of including the `alt` attribute within the `` element when you include an image in lieu of text; if a phone can't display the specified image, it will use the text defined in the `alt` attribute instead.

5.5.4 Icons

To include small images within text or alongside a menu, consider using the icons included with many phone models. Use the `localsrc` attribute within the `` element to specify an icon. (If the icon is not included within the phone's ROM, it is automatically uploaded from the server.)



Figure 5.9 Using icons

Some users might have devices that can present images, but have no icons available either in the device ROM or in its gateway server. You can still provide images to these phones by including both the `localsrc` and `src` attributes together within the `` element. In this case, the browser uses the `localsrc` attribute first, then the `src` attribute, then the `alt` attribute.

Figure 5.9 shows what happens when an icon is used, and the following example adds a reference for a left-hand icon:

```
<wml>
<card id="welcome">
<do type="accept">
  <go href="#search_by"/>
</do>
<p>
  <center>
    
    Movie lookup!<br/>
    <a href="#search by">
      <img src=http://www.domain.com/movie/splash.bmp
        alt="Movie Logo"/>
    </a>
  </center>
</p>
</card>
</wml>
```

```
</p>  
</card>  
</wml>
```

5.6 SUMMARY

This chapter discussed how to develop a menu from which the user can select an option, and how to develop an input form into which a user can enter data. Intrinsic to both of these topics is the assignment of a value to a variable. These values can either be associated with a menu selection or entered directly by the user.

Included in this chapter is a discussion on how images can be incorporated into a phone's display to replace or supplement text, enhance a user's experience, or to provide differentiated branding across an application.

Having learned techniques that will assign a value to a variable, the next step is learning to use the variables in a meaningful way. That is the topic of the next chapter.