

Building handheld applications with Waba

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

1. Introduction	2
2. Preparing to write Waba programs	5
3. Waba's class libraries and program structure	8
4. Building a simple Waba application	11
5. Compiling and generating code	15
6. Installing the application	18
7. Further resources	19

Section 1. Introduction

Introducing Waba

Handheld devices are everywhere. They are used for tasks as varied as scheduling appointments and checking e-mail on the road, to tracking consumables and schedules on construction projects. As more and more people acquire handheld devices, the demand for programming them is increasing.

This tutorial demonstrates how Waba allows the developer to build handheld applications using the Java language, the most popular object-oriented programming language, for two of the most popular types of handheld devices -- devices that run the Palm or Windows CE operating systems. Along the way we'll examine the tools for getting started on developing Waba applications, outline and explain the Waba classes, and develop a simple Waba application to provide concrete examples for the material discussed.

What Is Waba?

Waba is an open-source, Java-compatible platform designed to run on handheld devices. Waba, however, is not Java. Waba itself consists of a Java-compatible virtual machine (or VM) tailored to a particular type of handheld device (for example, a handheld device running the Palm OS) and a set of class libraries. Since Waba is Java compatible, it allows you to use your existing Java development tools to produce applications for handheld devices. It was initially developed to run on devices based on the Palm and Windows CE operating systems, although it has also been ported to other operating systems.

The advantages of Waba

Traditionally, writing applications for handheld devices required writing procedural C or machine code tailored to the particular device. It was very difficult to use modern object-oriented software development practices during development. It also required that the applications be written for a specific device. Because Waba is programmed with Java code, all of the same software development practices adopted for desktop applications can be applied to building handheld applications. Also, because Waba supports multiple handheld operating systems, your handheld application can be deployed on any device that supports Waba.

Differences between Waba and Java technology

While Waba allows you to program in Java code, it's important to recognize two important differences between the two platforms.

First, a Waba program only has access to the Waba class libraries. This means that a program cannot use the rich set of class libraries regularly available with Java 2 Standard Edition (J2SE). If you try to use any classes in a standard class library, your application will simply not run on a handheld device.

Second, the virtual machine under which Waba programs run -- the Waba VM -- is not identical to the J2SE Virtual Machine. For example, the Waba VM does not support floating point numbers. Use of the float or double data types in a Java program means the code cannot be directly ported to a handheld device.

These two facts combined make it impossible to simply take existing Java code, recompile it, and run it on a handheld device. Any existing Java code has to be ported to use only the Waba class libraries and avoid other J2SE VM features not supported by the Waba VM. Waba class libraries are discussed in detail later in the tutorial.

Waba versions and supported platforms

The developers of Waba, Wabasoft, have developed the Waba SDK to run under Microsoft Windows and the Waba VM under the Palm and Windows CE operating systems. Given that Waba is open source, a number of other developers have taken it upon themselves to port the Waba VM to additional devices. These devices include:

- Apple Newton
- MS-DOS
- Compaq IPaq
- TI Calculators

You can find a current listing of Waba VM ports on the Wabasoft Web site at <http://www.wabasoft.com>.

The Waba development process

Waba development is very similar to creating other Java applications. In order to develop Waba applications, you'll need:

- A Java development system running on your Microsoft Windows-based development machine
- The Waba SDK (available for download from the Wabasoft Web site)

You can use any Java development environment. Sun's Java Development Kit (JDK), Visual Café, and IBM Visual Age are all commonly used. This tutorial is based on the Sun JDK, which is available from [Sun's developer download site](#).

In order to run your application you'll also need a version of the Waba VM for your handheld device. This is also available from the Wabasoft Web site. Installation of the Waba SDK and VM is the topic of the next section.

Conventions used in this tutorial

There are a few conventions that are used in this tutorial to reinforce the material at hand:

- Text to be typed in is displayed in a **bold monospace** font. In some code examples, bold is used to draw attention to a tag or element being referenced in the accompanying text.
 - *Emphasis* is used to draw attention to windows, dialog boxes, and feature names.
 - A monospace font presents file and path names.
 - Throughout this tutorial, code segments irrelevant to the discussion have been omitted and replaced with ellipses (. . .).
-

About the author

Daniel A. Tauber, a principal in [Tauber Kienan Associates](#), is a technology strategist and solution architect. Dan brings a keen understanding of the drive for profitability and the need for developing customer-centric solutions to all initiatives. He has coauthored and contributed to numerous books on technology and Web management topics. His articles have covered development of high-end Java applications and open-source Java libraries. As a speaker or panel member, he has appeared at numerous conferences; he is also an instructor at the University of California Berkeley Extension and at San Francisco State University College of Extended Learning. Dan can be reached directly at dan@tauberkienan.com.

Section 2. Preparing to write Waba programs

What you need

Before developing Waba programs, you must download and install two packages. The first is the Waba Virtual Machine (VM). This is the Waba equivalent to the Java Virtual Machine (JVM) and is installed on the handheld device on which you'll run your programs. The second is the Waba SDK. The SDK contains the Waba class libraries and utilities to convert Java .class files to a format usable on the handheld device. This second package is installed on your development machine. In the next panel, we'll help you get those.

Downloading the Waba VM and Waba SDK

Both the Waba VM and Waba SDK are available for download from the Wabasoft Web site at <http://www.wabasoft.com/download.shtml>. For handheld applications destined for a Palm OS device, download the *Waba VM for Palm OS*.

If you're going to run your handheld applications on a Windows CE device, there are two VMs to choose from according to which processor architecture the destination device is based on: Waba VM for Windows CE MIPS or Waba VM for Windows CE SH3.

Regardless of which of the three versions of the Waba VM you select, you'll end up with a single ZIP file that contains the VM suited for your handheld device. Once you have the Waba VM downloaded, go to the [Wabasoft developers page](#) where you'll find the Waba SDK. The SDK consists of two ZIP files and is the same regardless of the target device.

Installing the SDK

To install the Waba SDK, unzip both files to the root of your development machine's C: drive. This creates a new folder named `wabasdk.10` containing the following subfolders:

- `bin` -- Utilities used to turn Java .class files into handheld executables
- `classes` -- The Waba class libraries
- `doc` -- The Waba class library documentation in HTML format

- `examples` -- A number of sample handheld applications
 - `src` -- Source code for the Waba class libraries
-

Installing the Waba VM

Installation procedures for the Waba VM vary according to the target device. The following section details the required steps for the two most common handheld platforms, Palm and Windows CE-based products. For instructions on installing the Waba VM on other supported platforms (for example, MS-DOS and Texas Instruments calculators) see the README files from the appropriate port. The [Online references](#) on page 19 section at the end of this tutorial also contains several useful Web links where you can find more information pertaining to alternate Waba platforms.

Installing the Waba VM for Palm OS

The Waba SDK on a Palm OS-based device consists of two files: `waba.pdb` and `waba.prc`. The Palm executable file `waba.prc` contains the actual Waba VM application. The Palm database file `waba.pdb` contains the Waba class libraries.

The installation procedure is relatively straightforward:

1. Unzip the Waba VM ZIP file that you downloaded from the Wabasoft Web site into a temporary directory.
 2. Double-click on both the `waba.prc` and `waba.pdb` files. This invokes the Install Tool dialog (the Install Tool is packaged with the Palm Desktop software) and places the name of the file in the list of programs to install during the next HotSync operation.
 3. Now simply HotSync the handheld. When the operation is complete, you can verify that the Waba VM installed correctly by tapping on the Waba VM icon in the Unfilled category; A welcome screen should appear.
-

Installing the Waba VM for Windows CE

Unlike Palm devices, which are based on a single processor type, Windows CE supports two different architectures, so be sure to download and install the correct Waba VM version. Once you've gathered the requisite files:

1. Extract the ZIP file to a temporary directory. The executable file `waba.exe` contains the Waba VM for your device. The file data file `waba.wrp` holds the Waba class libraries.
2. On your development machine, run the Windows CE Explorer (packaged with all Windows CE devices).
3. Within the Program Files folder on your Windows CE device, create a folder named `waba`; this folder will hold the Waba VM and class library.
4. Drag and drop `waba.exe` and `waba.wrp` from the temporary folder into the folder just created.

And that, as they say, is that. Now that you have the Waba SDK installed on your development machine and the Waba VM installed on your handheld device, you're ready to start programming. The next section looks at the various class libraries used by Waba and the structure of a Waba application. The examples and code segments discussed will eventually come together to form a simple handheld application that demonstrates a number of user interface elements.

Section 3. Waba's class libraries and program structure

The Waba class libraries

A Waba program cannot use the standard Java class libraries because they are not available for handheld devices. This means applications are limited to using the Waba class libraries. These class libraries provide a bridge between the handheld program and the handheld devices underlying the operating system and user interface. There are six libraries that ship with Waba:

- `waba.fx`
- `waba.io`
- `waba.lang`
- `waba.sys`
- `waba.ui`
- `waba.util`

The following panels discuss -- in broad strokes -- each of the above listed class libraries and their uses. Detailed information regarding these classes can be found in the documentation folder located in the Waba SDK directory.

waba.fx

The `waba.fx` library provides a set of classes and interfaces for dealing with 2D graphics, images, and sound. Keep in mind that the handheld device for which you are developing may have limitations when it comes to displaying certain colors, images, and/or fonts. For example, you can create a reference to the color red with the line:

```
red = new Color(255,0,0);
```

But you cannot actually display this color on a device with a gray-scale display such as the type used on most Palm-based devices.

Likewise, you can create `Font` objects referring to arbitrary fonts such as:

```
font = new Font ("Helvetica", Font.ITALIC, 10)
```

But, again, the above object does not display correctly on many devices.

waba.ui

This library contains a bridge between an application and the device's native user interface elements. It contains classes that represent visual elements such as buttons, labels, and windows. Much of the code written in your applications will typically be creating and manipulating classes from this library.

Waba suffers from the same problem as all cross-platform graphics libraries: It can only support the lowest common denominator across the supported platforms. This means that often you'll find native interface elements on the handheld device that cannot be accessed directly from a Waba application.

waba.lang

This is the most ingenious of the Waba class libraries. The `waba.lang` library corresponds to Java's special `java.lang` library. It contains the classes `Object`, `String`, and `StringBuffer`. Just as you never refer directly to the `java.lang` library when developing Java applications, you also never refer directly to the `waba.lang` library. Not only is this library automatically included in all Waba applications, the Waba VM automatically swaps `waba.lang` for Java's `java.lang` at run-time.

waba.sys

`waba.sys` contains classes related to the system itself. The current version includes three classes: `Convert`, `Time`, and `VM`.

`Convert` is used to convert data types between Waba's internal format and the handheld devices native format. You may need to use methods in this class if your program needs to interoperate with other handheld applications.

`Time` is a class that gives you access to the current date and time. Note that this class does not include any built-in methods for doing date arithmetic.

Finally, the static class `VM` allows your program to communicate with the Waba VM under which it is executing.

waba.util

The `waba.util` class is the smallest of all of the Waba class libraries. It only contains a single class: `Vector`. The `waba.util.Vector` class provides the same basic functionality as Java's `Vector` class. You can expect to see more utility classes added to this class as the Waba libraries continue to develop.

waba.io

The `waba.io` class contains functions for manipulating streams, files, networking, serial communications, and databases. Not all of the classes in this library work on all platforms that run Waba. For example, Palm OS devices do not support the notion of a file or file system, so using the `File` class on these devices generates an error.

Now that you have an overview of the Waba class libraries and VM, we're ready to begin building an application. The balance of this tutorial details what it takes to build a simple application that displays a message on the screen and responds to a button tap.

In building this application, you'll learn:

- The overall structure of a Waba application
- How to handle user interface elements
- How to respond to events triggered by user input

Section 4. Building a simple Waba application

The structure of a Waba application

All Waba applications share a common structure and are always derived from the `MainWindow` base class. User interface elements are added to the screen using a class constructor. Most Waba applications override the `onEvent` method, providing implementation tailored to the particular application as shown below.

```
import waba.ui.*;
class HandheldApp extends MainWindow {
    public HandheldApp () {
        // Initialization code goes here
    }
    public void onEvent (Event evt) {
        // Event handling code goes here
    }
}
```

In addition to `onEvent`, the `MainWindow` class has a number of other methods that can be overridden. These include:

- `onStart`: invoked when the application starts. As an alternative to creating the user interface elements in the constructor (as demonstrated above), you can also create them (as well as other initializations) using this method.
- `onExit`: invoked when the application exits. This is a useful place to put any clean-up code that has to be executed before your application shuts down.
- `onPaint`: invoked when the screen needs updating. If you are only using the built-in user interface elements (from the `waba.ui` library), you don't need to add any processing to update the screen -- it is done automatically. However, some forms of custom displays require you to use this method and provide your own screen-updating code.

Adding the UI code

With the basic skeleton of a Waba application in place, it's time to start adding application-specific code. In particular this application will create a few UI elements in the constructor shown in the following example.

```
private Label welcomeLabel;
private Button quitButton;
```

```
public HandheldApp () {
    int y=0;
    int deltaY = 18;

    welcomeLabel = new Label("Welcome to our Waba demonstration application.");
    welcomeLabel.setRect(5, 5, this.width-5, deltaY);
    add(welcomeLabel);
    y+=deltaY;

    quitButton = new Button("Quit");
    quitButton.setRect((this.width/2) - 20, this.height-15, 40, 15);
    add(quitButton);
}
```

As you can see, each user interface element has a corresponding Waba class. The `Button` class (shown in blue) corresponds to a button and the `Label` class (shown in red) corresponds to a text label on screen. After creating each object, they are positioned by invoking the `setRect` method. Finally, they are added to the screen by invoking the `MainWindow` object's `add` method. This same pattern is followed when adding any type of user interface element supported by Waba.

Responding to user input

All input coming from the user is dispatched to the `onEvent` method and handled by overriding this method with code to respond to the events. For example, the following code snippet responds to a tap on the Quit button by terminating the application.

```
public void onEvent (Event evt) {
    if (evt.type==ControlEvent.PRESSED) {
        if (evt.target==quitButton) {
            exit(0);
        }
        // other event handling code can go here
    }
}
```

All events are represented by classes derived from `Event`. The Waba classes derived from `Event` are `ControlEvent`, `KeyEvent`, and `PenEvent`. Each of these classes can represent any one of a number of different events.

ControlEvent

The user interacting with a class derived from `Control`, such as `Button`, generates a control event. You can determine what user action caused the event by inspecting `Event.type`. The possible control event types are:

- `ControlEvent.PRESSED` triggered by the user tapping on the control
 - `ControlEvent.FOCUS_IN` triggered by the user giving the control focus
 - `ControlEvent.FOCUS_OUT` triggered by the user moving the focus to a different control
 - `ControlEvent.TIMER` triggered when a timer created with `Control.addTimer` ticks
-

KeyEvent

Key events are generated whenever the user enters input via a keyboard or other input method (such as graffiti). For this event, `KeyEvent.event` will always equal `KeyEvent.KEY_PRESSED` because this is the only type of user action that triggers this event.

PenEvent

Finally, pen events are triggered when the user drags the stylus around the handheld device's screen.

- `PenEvent.PEN_DOWN` is triggered when the user presses the stylus against the handheld device's screen.
 - `PenEvent.PEN_UP` is triggered when the user removes the stylus from the screen.
 - `PenEvent.PEN_DRAG` is triggered when the user drags the stylus around the screen.
 - `PenEvent.PEN_MOVE` is triggered when the user moves the stylus around the screen.
-

The complete program

Here is the complete program:

```
import waba.ui.*;

class HandheldApp extends MainWindow {

public HandheldApp () {
```

```
        int y=0;
        int deltaY = 18;

        welcomeLabel = new Label("Welcome to our Waba demonstration application.");
        welcomeLabel.setRect(5, 5, this.width-5, deltaY);
        add(welcomeLabel);
        y+=deltaY;
        quitButton = new Button("Quit");
        quitButton.setRect((this.width/2) - 20, this.height-15, 40, 15);
        add(quitButton);
    }

    public void onEvent (Event evt) {
    if (evt.type==ControlEvent.PRESSED) {
        if (evt.target==quitButton) {
            exit(0);
        }
    }
    }
}
```

Save this file to HandheldApp.java. Next up: Compiling the Waba application.

Section 5. Compiling and generating code

Compiling with the JDK

Waba applications are compiled just like any other Java application, except you must override the regular Java class libraries with the Waba class libraries. The Waba class libraries are included along with the Waba SDK.

To compile the HandheldApp program with the Sun JDK, enter:

```
javac -classpath c:\wabasdk.10\classes;. HandheldApp.java
```

When the compiler process completes, you should have a new file in your local directory called `HandheldApp.class`.

Running your application

You can run the Waba application on a Window-based machine using the viewer, which is part of the Waba SDK. To view the program on a PC, enter:

```
java -classpath c:\wabasdk.10\classes;. waba.Applet.applet HandheldApp
```

A window appears showing the application. While it's not possible to fully test it from a desktop machine, you can get a good sense of how the application will look and behave.

Turning .class files into executable files: Creating the executable

You cannot simply install the `.class` file generated with the Java compiler on a Palm OS or Windows CE based device. You'll have to process the `.class` file with a special program that turns it into an executable and data file specific to the device on which it will run. The process is identical for both Palm- and Windows CE-based systems, although the files generated for each platform are different.

The first step in building the files needed for the handheld device is to process the `.class` file with the `exegen` program that comes with the Waba SDK. This program

reads in the `.class` file and generates a `.prc` file for Palm OS-based handhelds and a `.lnk` file for Windows CE-based handhelds. The second step is to package the class files required by your application into a data file appropriate for the handheld device.

You use the `exegen` program to create `.prc` and `.lnk` files. To run this program, enter:

```
exegen HandheldApp HandheldApp HandheldApp
```

More on `exegen`

The `exegen` command line displayed in the previous panel looks odd because it repeats the same name three times (actually you could use different names for each one, but that makes the whole file naming process much more confusing). The first `HandheldApp` is the name of the handheld executable. The second `HandheldApp` is the name of the class that contains the application. The third `HandheldApp` is the name of the file that contains the classes on the handheld device.

`exegen` has a number of other features, including the ability to include an icon for the program that will be displayed on the handheld device. Run `exegen` without any parameters for a full list of options.

Also, be warned that `exegen` is a very popular program name. In addition to the `exegen` that comes with the Waba SDK used throughout this tutorial, Microsoft has a program called `exegen` that is used to convert Java `.class` files into Windows executables, and Sun has a version that is used to convert Java `.class` files into executables for handheld devices using *J2ME*. If you use more than one vendor's SDK, it's a good idea to always specify a full pathname to the `exegen` executable.

Packaging the `.class` Files

Once the executable is built, the `.class` files need to be packaged into a format appropriate for your handheld device. This is done using the `warp` utility that comes with the Waba SDK. To do this, enter

```
warp c HandheldApp HandheldApp.class
```

The `c` parameter tells `warp` that you are creating a new file. `HandheldApp` is the name of the Palm database and Windows CE data file to be created. Finally, `HandheldApp.class` is the name of the Java class file that we're going to package

into the Palm database file. Note: If your handheld application consists of more than one `.class` file (which means you have more than one `.java` source file), you'll simply list each of the `.class` files on the `warp` command line. All `.class` files that make up your application must be packaged into a single data file.

You should now have four new files in your working directory. `HandheldApp.prc` is the Palm executable file, `HandheldApp.lnk` is the Windows CE executable file, `HandheldApp.pdb` is the Palm data file, and `HandheldApp.wrp` is the Windows CE data file. The application is now built and ready to be installed on your handheld device.

Section 6. Installing the application

Palm OS or Windows CE?

Once packaged for a handheld device, the Waba application is just like any other application for that device, and is installed just as any other would be. Of course, the method varies depending on the type of device you're installing to.

Installing on the Palm OS

Installing the application on a Palm OS device is a snap. Just double click on the two Palm files created in the last section, `HandheldApp.prc` and `HandheldApp.pdb` in turn. The Install Tool window should appear with the file listed; just click the Done button. Next time you HotSync your device, the application is installed.

After the program has been installed, it will appear in the *Unfiled* category. You can run it by simply tapping on its icon.

Installing on a Windows CE device

To install the application on a Windows CE-based device, copy the two Windows CE files created by `exegen` and `warp` to the handheld device.

1. On your development machine, run the Windows CE Explorer.
2. Within the Program Files folder on your Windows CE device, create a `HandheldApp` folder. This is the folder that will hold the `HandheldApp` class file.
3. Drag and drop `HandheldApp.wrp` from your development folder into the `HandheldApp` folder.
4. Drag and drop `HandheldApp.wrp` from your development folder into the `HandheldApp` folder.
5. Drag and drop the `HandheldApp.lnk` file from your development folder into the handheld's Start folder.

You can now run your `HandheldApp` application by selecting it from the handheld's Start menu.

Section 7. Further resources

Online references

Waba is a powerful tool for building handheld applications. This tutorial only touched on some of Waba's abilities for creating handheld applications. Several great places on the Web to continue your education about Waba include:

- <http://www.wabasoft.com> is the official home of Waba. This is where you'll find the most up-to-date information about Waba and Waba development, as well as the latest version of the Waba SDK and VM.
- <http://www.superwaba.org> is a site dedicated to an extension of Waba with support for color and other features.
- <http://www.wabaworkbench.com> is a portal dedicated to information about building Waba applications. Here you'll find articles, links to different Waba VMs, utilities, and sample code.
- <news://news.falch.com/pilot.programmer.waba> The pilot.programmer.waba newsgroup is the official gathering spot to discuss and ask questions about Waba.
- <http://www.thisiscool.com/doswaba.htm> is the official home of the MS-DOS port of Waba.

Feedback

Please send us your feedback on this tutorial. We look forward to hearing from you!

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT stylesheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)

You can get the source code for the Toot-O-Matic at www6.software.ibm.com/dl/devworks/dw-tootomatic-p. The tutorial [Building tutorials with the Toot-O-Matic](#) demonstrates how to use the Toot-O-Matic to create your own tutorials. developerWorks also hosts a forum devoted to the Toot-O-Matic; it's available at www-105.ibm.com/developerworks/xml_df.nsf/AllViewTemplate?OpenForm&RestrictToCategory=11.

We'd love to know what you think about the tool.