

"If you're interested in developing for this burgeoning platform, there is no one better able to get you up-to-speed."

—From the Foreword by **Rob Tiffany**,
mobility architect, Microsoft

Second Edition



Programming .NET Compact Framework 3.5



Paul Yao
David Durant

Praise for *Programming .NET Compact Framework 3.5, Second Edition*

“Each day it seems that mobile devices are becoming more common and critical to our everyday lives. As more developers target this growing market, the need for a trusted guide grows, and this book provides that guidance. The authors have reached a rare balance in which they provide the core knowledge required to start the journey of mobile development as well as the depth to support a seasoned developer. Every developer building applications for Windows Mobile devices should have this book close at hand to guide them through the challenges and rewards of creating their applications.”

—Mike Erickson, *principal consultant, Neudesic*

“This is one of the most complete technical books available on developing smart-device applications with Visual Studio and the Compact Framework 3.5. It covers many necessary topics, from preserving battery life to communicating with the outside world using WCF and everything in between. Whether you have been developing in the Compact Framework for years or are just getting started, this book has something in it for you.”

—Jason Estes, *senior software engineer, Itron*

“This book arms Windows Mobile developers with the core knowledge they need for building rich business and personal applications. It also gives them great insight into what is going on under the covers, how the platform is unique compared to other versions of Windows, and how to build robust Windows Mobile applications productively.”

—Brian Noyes, *chief architect, IDesign, Inc.*

“This is the best resource I’ve ever seen for programming the Compact Framework. This book should be on the desk of everybody doing mobile development.”

—Rebecca M. Riordan, *author of Seeing Data*

“With valuable information added to this new edition, Paul Yao and David Durant have enhanced their already invaluable guide to programming for the .NET Compact Framework. If you’re a serious mobile programmer, this is the one book you need on your shelf—but it won’t stay there.”

—Josh Trupin, *Microsoft Corporation*

“Yao and Durant’s second edition of their popular .NET CF book is a much-needed update for all developers who want to build apps for the Windows Marketplace for Mobile. The book provides comprehensive coverage of the technology, and it is well suited for both .NET CF beginners and experts.”

—Michael Yuan, *cofounder, Ringful LLC*

Programming .NET Compact Framework 3.5

Table of Contents

Contents

Figures

Tables

Foreword

Preface

Acknowledgments

About the Authors

1 Mobile Phone Programming

1.1 Selecting an Application Programming Interface

1.1.1 The Win32 API

1.1.2 The .NET Compact Framework

1.1.3 The Web Browser

1.1.4 Rich Internet Applications (RIAs)

1.2 Memory Management

1.2.1 Metadata Maps

1.2.2 The JITted Code Pool

1.2.3 Garbage Collector Pools

1.2.4 Garbage Collection and Data

1.2.5 Automatic Garbage Collection

1.2.6 Special Handling of Managed Data

1.2.7 Manual Memory Management of Native Data

1.3 Conclusion

Table of Contents

2 Extending Battery Life

2.1 What Is the Problem?

2.1.1 Basic Questions to Ask

2.1.2 Factors Influencing Battery Life

2.2 Measuring Battery Usage

2.2.1 Talk Time and Standby Time

2.2.2 Power Measurement Method 1: Software-Only Approach

2.2.3 Power Measurement Method 2: Dedicated Hardware

2.3 Device Power Study

2.3.1 Standby Power Usage

2.3.2 Backlight Power Usage

2.3.3 Power Usage for Communicating

2.3.4 Multimedia Power Usage

2.4 Conclusion

3 Platform Invoke

3.1 Overview of P/Invoke

3.1.1 When to Use P/Invoke

3.1.2 Why We Prefer .NET Compact Framework Classes over Win32 Functions

3.1.3 Porting Native Code to Managed Code

3.1.4 Component Object Model (COM) Support

3.2 Creating P/Invoke Declarations

3.2.1 A Simple Function: MessageBox

3.2.2 Native Function Details

3.2.3 Function Return Values

3.2.4 Getting Started: C-Style Function Declarations

3.3 Supported P/Invoke Function Parameters

3.3.1 The Limits of Parameter Passing

3.3.2 Simple Data Types

Table of Contents

3.3.3 Passing Parameters by Value versus by Reference

3.3.4 Passing String Parameters by Value

3.3.5 Structures

3.3.6 Type of Types versus Type of Parameters

3.3.7 Arrays

3.4 A Sample Program: CallWin32

3.5 Writing Win32 Dynamic Link Libraries

3.5.1 Declaring C++ Functions in a DLL

3.6 Manual P/Invoke Parameter Passing

3.6.1 The Marshal Class

3.6.2 Copying to Native Memory

3.6.3 Fine-Tuning Structures with the MarshalAs Attribute

3.7 Communicating between Native and Managed Code

3.7.1 The MessageWindow Class

3.7.2 Other Ways to Communicate between Native and Managed Code

3.8 Comparing P/Invoke Support

3.8.1 Windows CESpecific Differences

3.8.2 .NET Compact Framework Implementation Details

3.9 Conclusion

4 Data Binding to Controls

4.1 Data Binding

4.1.1 Data-Bindable Controls

4.1.2 Data-Bindable Objects

4.2 Complex Data Binding

4.2.1 Using Complex Data Binding with ComboBox Controls

4.3 Simple Data Binding

4.3.1 The BindingsCollection Property

4.3.2 Formatting and Parsing

4.4 The DataGrid Control

Table of Contents

- 4.4.1 Using Complex Data Binding with the DataGrid Control
- 4.4.2 Styling the Display of Data in a DataGrid Control
- 4.4.3 Creating Table and Column Styles
- 4.4.4 Creating Styles at Runtime
- 4.4.5 Responding to User Input
- 4.4.6 Using Simple Data Binding with the DataGrid Control
- 4.4.7 Accessing DataGrid Information
- 4.4.8 Providing Drill-Down Capability
- 4.4.9 Providing In-Place Editing Capability
- 4.4.10 Providing Automated In-Place Editing Capability

4.5 Conclusion

5 Storage

5.1 Smart-Device Data Storage

- 5.1.1 Installable File Systems
- 5.1.2 The Windows CE File System
- 5.1.3 ROM-Based Files

5.2 File I/O

- 5.2.1 The File and Directory Classes
- 5.2.2 Byte-Level I/O
- 5.2.3 Higher-Level I/O
- 5.2.4 Encoding and Decoding Data
- 5.2.5 Using the I/O Classes
- 5.2.6 Text File I/O
- 5.2.7 Binary File I/O
- 5.2.8 Writing Binary Data
- 5.2.9 Reading Binary Data
- 5.2.10 Binary I/O and Structures
- 5.2.11 XML File I/O
- 5.2.12 Easier XML Serialization

5.3 Registry Access

Table of Contents

5.3.1 Opening and Creating Registry Keys

5.3.2 Reading and Writing Registry Values

5.3.3 Updating the Storage Sample Application to Use the Registry

5.4 Conclusion

6 ADO.NET Programming

6.1 Examining ADO.NET

6.1.1 A Layered Approach

6.1.2 The ADO.NET Classes

6.1.3 ADO.NET Error Handling

6.2 Working with Data Sets

6.2.1 Creating and Accessing DataSet, DataTable, and DataView Objects

6.2.2 Data Binding

6.2.3 Reading and Writing a Data Set as XML

6.3 Microsoft SQL Server CE

6.3.1 SQL Server CE Files

6.3.2 SQL Server CE Syntax

6.3.3 SQL Server CE Query Analyzer

6.3.4 Creating a SQL Server CE Database

6.3.5 Populating a SQL Server CE Database

6.3.6 Retrieving and Displaying Data

6.3.7 Updating a SQL Server CE Database

6.3.8 The SqlCeDataAdapter Class

6.3.9 Querying Schema Information

6.4 Microsoft SQL Server

6.4.1 Connecting to SQL Server

6.4.2 Creating Command Objects

6.4.3 Using SQL Server Stored Procedures

6.4.4 Using Stored Procedures with DataSet Objects

6.4.5 DataSet Objects and Concurrency

Table of Contents

6.4.6 Using Multi SELECT Stored Procedures

6.4.7 Working with Typed Data Sets

6.5 Web Services

6.5.1 XML, XSD, and SOAP

6.5.2 A Web Services Application

6.5.3 A Web Services Client Application

6.6 Conclusion

7 LINQ

7.1 Overview

7.1.1 Set Classes in .NET

7.1.2 LINQ in the Compact Framework

7.1.3 Deferred Execution

7.2 The Sample Application

7.2.1 LINQ to Datasets

7.2.2 LINQ to Objects

7.2.3 Business Object Properties

7.2.4 The Hybrid Version

7.3 LINQ to XML

7.4 Conclusion

8 Synchronizing Mobile Data

8.1 Understanding SQL Server CE Synchronization

8.1.1 Three Synchronization Mechanisms

8.1.2 IIS Connectivity

8.1.3 Database Connectivity

8.2 Installing Remote Data Connectivity

8.2.1 Creating the Virtual Directory

8.2.2 Configuring Additional Components

8.3 Using RDA

8.3.1 RDA Capabilities and Overhead

Table of Contents

8.3.2 Programming for RDA

8.4 Using Merge Replication

8.4.1 Using Good Design to Avoid Synchronization Failures

8.4.2 Configuring Merge Replication

8.4.3 Programming for Merge Replication

8.5 Choosing between Merge Replication and RDA

8.6 Using Data Synchronization Services

8.6.1 Understanding Data Synchronization Service Requirements

8.6.2 Building a Data Synchronization Service

8.7 Conclusion

9 The Remote API

9.1 RAPI Fundamentals

9.1.1 Available RAPI Functions

9.1.2 Building .NET ActiveSync Applications

9.1.3 RAPI Startup and Shutdown

9.2 Accessing the Object Store

9.2.1 Using RAPI to Access Device Files

9.2.2 Remote Access to Device Registry Entries

9.2.3 Remote Access to Device Property Databases

9.3 Detecting Changes in Device Connection State

9.3.1 The Auto-Start Approach

9.3.2 The Callback Approach

9.4 Loading Programs and DLLs

9.4.1 Running Device-Side Programs

9.4.2 Loading Device-Side DLLs

9.5 Conclusion

10 Windows Communication Foundation

10.1 What Is WCF?

Table of Contents

10.1.1 WCF Terminology

10.1.2 WCF in the .NET Compact Framework

10.2 Creating a WCF Service

10.2.1 Generating the Code

10.2.2 Making a Windows MobileCompatible WCF Service

10.2.3 Setting the Host Address

10.2.4 SmartMeter: A Sample WCF Service

10.3 Creating a WCF Client in Windows Mobile

10.3.1 WCF Client Namespaces and Assemblies

10.3.2 Generating the WCF Client Proxy

10.3.3 Instantiating a WCF Client

10.3.4 Accessing the WCF Service

10.3.5 WCF Client Sample: ReadSmartMeter

10.4 Conclusion

11 Creating Graphical Output

11.1 An Introduction to .NET Compact Framework Graphics

11.1.1 Drawing Surfaces

11.1.2 Drawing Function Families

11.1.3 .NET Compact Framework Graphics

11.2 Drawing on the Display Screen

11.2.1 Accessing a Graphics Object

11.2.2 Drawing in Controls

11.2.3 The Paint Event

11.2.4 Non-Paint Event Drawing

11.3 Raster Graphics

11.3.1 Specifying Colors

11.3.2 Creating Brushes

11.3.3 Creating Bitmaps

11.3.4 Drawing Bitmaps

Table of Contents

11.3.5 A Sample Program: ShowBitmap

11.4 Vector Graphics

11.4.1 Creating Pens

11.4.2 A Game: JaspersDots

11.5 Conclusion

12 Text and Fonts

12.1 Drawing Text

12.1.1 Text-Drawing Support in the .NET Compact Framework

12.1.2 The DrawString Method

12.1.3 A Sample Program: SimpleDrawString

12.2 Font Selection

12.2.1 The Font Property of Controls

12.2.2 Enumerating Fonts

12.2.3 A Sample Program: FontPicker

12.2.4 A Sample Program: RotateText

12.3 Placing Text

12.3.1 Text Size and the MeasureString Method

12.3.2 A Sample Program: MeasureString

12.3.3 Text Alignment

12.3.4 A Sample Program: TextAlign

12.4 Conclusion

A: Hungarian Notation for .NET Programs

A.1 Goals and Objectives

A.2 Guidelines

A.3 .NET Naming Guidelines

A.4 Hungarian Notation

A.4.1 Our Use of Hungarian Notation

A.4.2 The m_ Prefix for Private Data

A.4.3 Hungarian Prefixes for CTS Value Types

Table of Contents

B: Windows API Allocation and Cleanup Functions

Index