

Module 1: Introduction to Mobile Device Application Development

Contents

Overview	1
Lesson: Platforms, Tools, and Technologies	2
Lesson: Overview of the .NET Compact Framework	8
Lesson: Introduction to Smart Device Extensions	14
Review	23



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2002 Microsoft Corporation. All rights reserved.

Microsoft, MS-DOS, Windows, Windows NT, *<plus other relevant MS trademarks, listed alphabetically. The publications specialist replaces this example list with the list of trademarks provided by the copy editor. Microsoft, MS-DOS, Windows, and Windows NT are listed first, followed by all other Microsoft trademarks listed in alphabetical order.>* are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

<The publications specialist inserts mention of specific, contractually obligated to, third-party trademarks, provided by the copy editor>

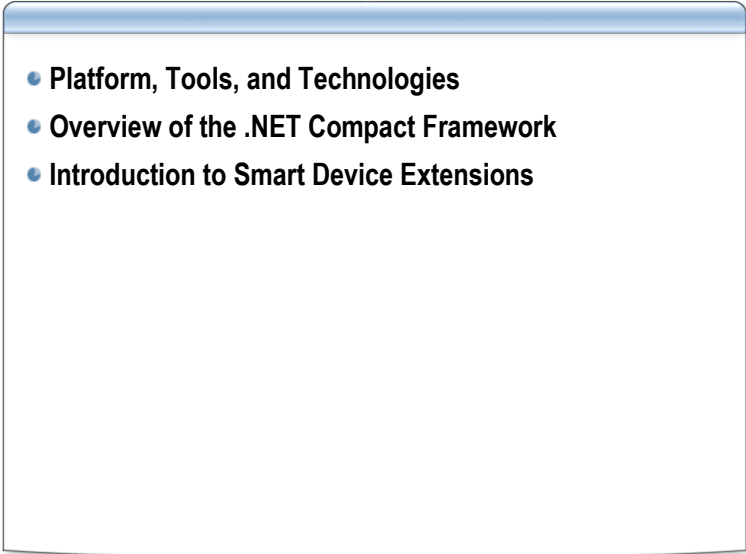
The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Instructor Notes

[Instructor_notes.doc](#)

Overview

Overview

- 
- Platform, Tools, and Technologies
 - Overview of the .NET Compact Framework
 - Introduction to Smart Device Extensions

Introduction

The Microsoft® .NET Compact Framework brings managed code and XML Web services to smart device application development. Although the .NET Compact Framework supports a rich subset of the .NET Framework, and thus provides the same benefits as the .NET Framework, it is designed specifically for resource-constrained devices, such as personal digital assistants (PDA), set-top boxes, and smart mobile phones.

Smart Device Extensions (SDE) extend Microsoft Visual Studio® .NET for device development by building on the existing rapid development features of Visual Studio .NET. Developers now have a single application development environment for device, desktop, and server.

In this module, you will learn about the platforms and development tools that are available as you develop mobile device applications using the .NET Compact Framework. You will gain an understanding of how the .NET Compact Framework greatly simplifies the process of creating and deploying applications to mobile devices while also enabling you to take full advantage of the capabilities of the device.

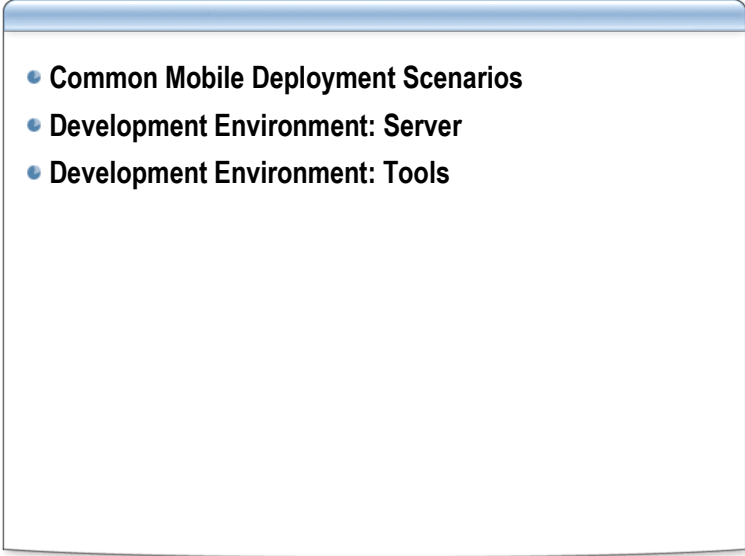
Objectives

After completing this module, you will be able to:

- Describe common scenarios in a mobile enterprise solution.
- Describe the role of the .NET Compact Framework and Smart Device Extensions in the development of mobile device applications.
- Identify key difference between the .NET Compact Framework and the full .NET Framework.
- Identify the server and client tools that are required in a mobile application development environment.
- Configure the Pocket PC 2002 Emulator.

Lesson: Platforms, Tools, and Technologies

Lesson: Platforms, Tools, and Technologies

- 
- Common Mobile Deployment Scenarios
 - Development Environment: Server
 - Development Environment: Tools

Introduction

This lesson describes common mobile enterprise scenarios and highlights typical development challenges and issues that occur when integrating mobile users and mobile devices into the enterprise computing environment. You will learn how mobility is being used today in the enterprise to extend business efficiency, improve communication, and provide access to mission critical data.

This lesson covers the development environment, and outlines the components that are located on the server and the client.

Lesson objectives

After completing this lesson, you will be able to:

- Explain how devices fit into a mobile enterprise solution.
- Identify mobile development options.
- Describe common scenarios in a mobile enterprise environment.
- Identify the server products required in a mobile enterprise application development project.
- Identify different mobile application development tools.
- Describe the Smart Device Extensions component of the Visual Studio .NET environment.

Lesson agenda

This lesson includes the following topics:

- Common Mobile Device Scenarios
- Development Environment: Server
- Development Environment: Tools

Common Mobile Deployment Scenarios

Common Mobile Deployment Scenarios

- **Integrating devices with a multitier desktop environment**
 - TCP/IP, HTTP, XML, SOAP, and XML Web services
 - Security: user authentication and data encryption
 - Access through firewalls
- **Offline vs. online**
 - Device detached from the network
 - Data cached locally for use offline
 - Intelligent synchronization of data when connected
 - Wireless connectivity

Introduction

Because of the explosive growth of wireless devices connected to networks over the last five years, organizations are discovering that intelligent devices are a natural extension of the enterprise. For example, the healthcare, retail, and real estate industries have relied heavily on mobile devices for some time.

Integration

The Internet/intranet is an accepted enterprise computing component of most large and medium organizations. Like personal computers, mobile devices also use established communication standards, such as TCP/IP, HTTP, XML, and SOAP.

In particular, XML Web services provide a lightweight distributed computing environment that extends easily to mobile clients, works across organizational boundaries, and handles devices outside the firewall.

Security is critically important in an enterprise because of the variety of connectivity options in a mobile scenario. An organization must carefully consider how to secure access to a device to prevent theft and loss, how to authenticate users, and how to encrypt data and communication links.

Device connectivity

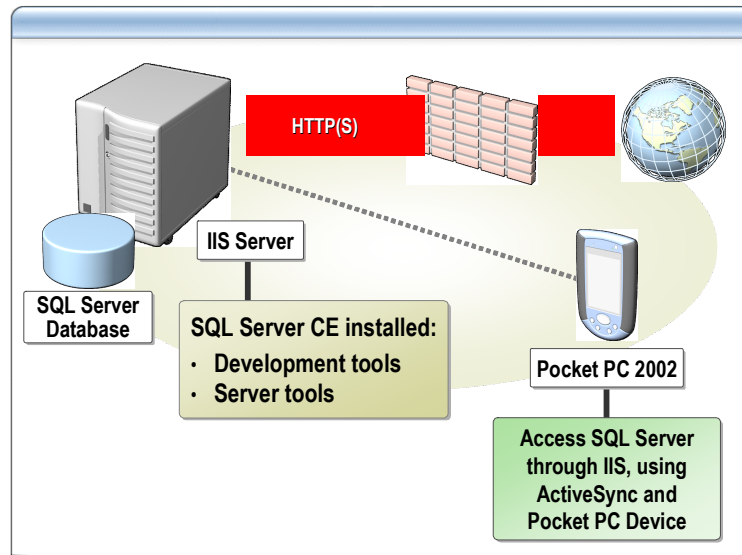
Application mobility involves more than simple wireless connectivity. While truly mobile applications are able to go anywhere, the connection state may be transparent to the user.

A device that is capable of running application code locally, such as a Pocket PC, Smartphone, or any programmable Microsoft Windows® Powered device, is suited to a connectivity model where a user downloads data for offline use. The user later reconnects to the network and updates the local data with the data source.

Alternatively, a Web application, with Web pages that are formatted to run on small form factor devices, must remain continuously connected because the application code runs on the server

Development Environment: Server

Development Environment: Server



Introduction

In a single-server environment, all systems are on one computer. In a multiple-server environment, systems are on many different computers. You can combine a single-server and development environment on one computer to create a complete development and test environment.

Server environment

The server-side environment includes the following:

- Microsoft SQL Server
- Microsoft Internet Information Services (IIS)

Microsoft SQL Server™ 2000 Windows CE Edition (SQL Server CE) extends Microsoft SQL Server to Microsoft Windows Powered mobile devices. You must also configure the server running Microsoft Internet Information Services (IIS) for SQL Server CE, by installing the SQL Server CE 2.0 Server Tools.

Before you install SQL Server CE, you must install the following software on the development system:

- Microsoft Visual Studio .NET
- Microsoft ActiveSync® 3.1 or later

Microsoft ActiveSync

You use ActiveSync in the development environment to deploy and debug applications. In the client environment, mobile devices use ActiveSync 3.1 in conjunction with SQL Server CE Relay if the Windows Powered device uses a serial, infrared (IR), or USB connection to communicate to SQL Server by using IIS through the network connection on a desktop computer

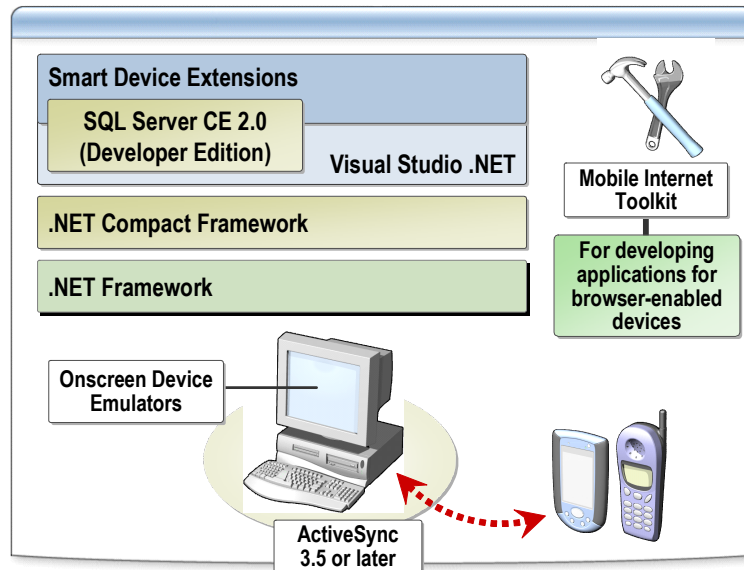
Important Pocket PC 2002 devices using ActiveSync 3.5 do not require SQL

Server CE Relay, because ActiveSync 3.5 supports Ethernet connectivity to and from the device through the desktop computer

For more information about installing and configuring SQL Server CE, see SQL Server CE Books Online.

Development Environment: Tools

Development Environment: Tools



Introduction

The .NET Compact Framework, a rich subset of the full .NET Framework, is at the heart of Smart Device Extensions (SDE) for Visual Studio .NET. The .NET Framework provides a single programming model for programming the server and the desktop computer. The .NET Compact Framework extends this model to smart, resource constrained devices.

Smart Device Extensions

Microsoft Visual Studio .NET is the primary development tool used when creating mobile device applications. The .NET Compact Framework is the runtime platform for your application development system for Windows Powered mobile devices. Smart Device Extensions for Visual Studio .NET, allows you to design, debug, and deploy applications on many of today's mobile devices.

If you have already developed desktop applications by using Visual Studio .NET, you can use your existing knowledge to create applications for the Pocket PC or Windows CE .NET powered devices. Visual Studio .NET provides an integrated suite of application development tools, regardless of whether you are targeting the mobile device or the desktop computer. These shared tools include:

- Choice of Object Oriented languages (Microsoft Visual Basic® or C#)
- Visual Forms Designer for creating rich user interfaces
- ADO.NET for database access
- Visual Database Designer
- Component Designers for visual and nonvisual control creation
- XCOPY application deployment

Smart Device Extensions also include several high fidelity, onscreen device emulators. Emulators are provided for Pocket PC, Pocket PC 2002, Smart phones, and devices running Windows CE .NET.

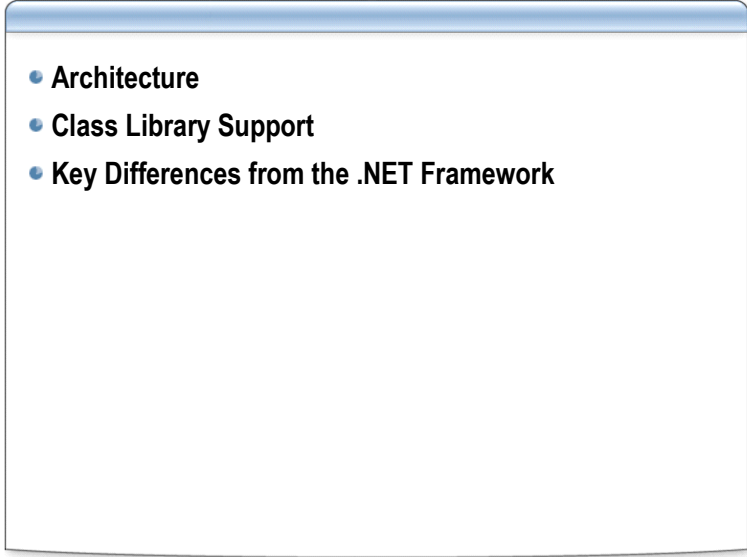
Running applications locally vs. on the server

There are two different approaches to programming mobile devices. You can place the application on the device, or you can place the code on the server and present the user interface through a browser. The .NET Compact Framework facilitates the former, and the Microsoft Mobile Internet Toolkit (MMIT) facilitates the latter.

Building mobile applications that run code on the device allows an application to continue to run while disconnected from the network. Building mobile applications where the code runs mostly on the server also has its advantages, particularly when you want to target the widest range of Internet-enabled mobile devices. MMIT complements Smart Device Extensions and the .NET Compact Framework.

Lesson: Overview of the .NET Compact Framework

Lesson: Overview of the .NET Compact Framework

- 
- Architecture
 - Class Library Support
 - Key Differences from the .NET Framework

Introduction

The .NET Compact Framework is a subset of the full .NET Framework, that provides many of the same benefits as the full .NET Framework. Some of these benefits include: the common language runtime, just-in-time (JIT) compilation, evidence-based code security, memory management through garbage collection, Windows Forms, data access, XML, and the ability to consume XML Web services.

This lesson provides a brief overview of the .NET Compact Framework and a comparison with the full .NET Framework.

Lesson objectives

After completing this lesson, you will be able to:

- Explain the .NET Compact Framework architecture and compare it to the full .NET Framework.
- Describe key differences between the .NET Framework and the .NET Compact Framework.
- Identify the class library support in the .NET Compact Framework at the namespace level.

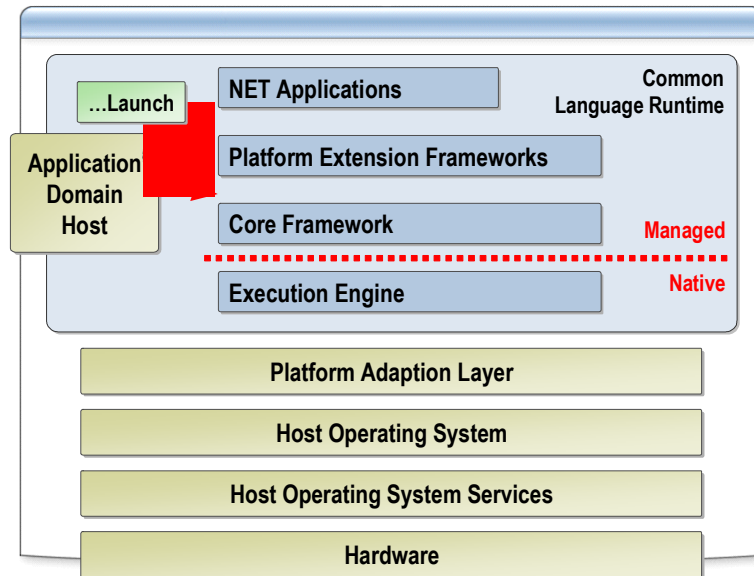
Lesson agenda

This lesson includes the following topics:

- Architecture
- Class Library Support
- Key Differences from the .NET Framework

Architecture

Architecture



Introduction

The .NET Compact Framework and the full .NET Framework share much in common. However, platform size and performance characteristics dictate that the .NET Compact Framework architecture differs somewhat from the full version. The .NET Compact Framework is the lightweight version of the .NET Framework, designed for use on resource-constrained devices.

Application domains

Every .NET Compact Framework application runs inside a runtime construct called an application domain, which is similar to an operating system process. The application domain host, itself a native application, starts an instance of the common language runtime for running managed code. You can run managed and native applications concurrently. You can also selectively invoke native APIs of a device operating system.

Note The .NET Compact Framework provides operating system and CPU portability. You do not need to recompile applications for each CPU that runs the .NET Compact Framework.

Memory management

The .NET Compact Framework is optimized for battery-powered systems and avoids heavy use of random access memory (RAM) and CPU cycles. Because devices may not contain hardware memory management units (MMU) or operating system virtual memory, the .NET Compact Framework must use available resources efficiently.

Garbage collection

The .NET Compact Framework ensures that all managed resources used by a running application are freed or returned to the host operating system when the application ends. The .NET Compact Framework itself should not fail because of low memory.

The .NET Compact Framework determines when garbage collection should be run. Garbage collection can occur in a single application domain or in all application domains. This prevents one application domain from using too much memory at the expense of others.

Storage

The .NET Compact Framework uses available RAM or read-only memory (ROM) to enable applications to run in low memory situations. If RAM storage is not available, or not sufficient, the use of ROM, flash memory, or disk storage allows an application to continue to run with decreased performance.

The .NET Compact Framework uses available RAM, up to a limit specified by the device, to store dynamic data structures and JIT-compiled code and then frees the memory when appropriate.

The common language runtime uses a code-pitching technique to free blocks of JIT-compiled code at run time when memory is low. This helps larger applications to perform satisfactorily on RAM-constrained systems.

The following items can be stored in either RAM or ROM:

- Native code that forms the common language runtime
- Files that contain Microsoft Intermediate Language (MSIL) instructions and metadata for class libraries

Class Library Support

Class Library Support

System			
System.Windows.Forms	System.Drawing	System.Data	System.Xml
Design	Drawing2D	ADO.NET	XmlDocument
ComponentModel	Text	SqlClient	Readers/Writers
		SqlServerCe	
		System.Web	System.Web.Services
		UI	Description
		Services	Discovery
		Security	Protocols
System			
Collections	IO	Security	Net
Text	Reflection	Diagnostics	Globalization
Resources	Threading		

Introduction

.NET Framework classes and class members are organized hierarchically under namespaces. The .NET Compact Framework shares approximately two thirds of the classes in common with the full .NET Framework. However, you should be aware of some important differences.

Classes shared by the Frameworks

The .NET Compact Framework provides you with a rich subset of the classes and class methods to develop applications that run on resource-constrained devices. Those classes are semantically compatible with same-named classes in the .NET Framework. In addition to the common language runtime, the .NET Compact Framework includes class support for important features like:

- XML Web services consumption
- Relational data management (local or remote)
- Rich XML functionality
- Rich drawing classes, such as image controls.
- Powerful user interface creation

The .NET Compact Framework does not support ASP.NET classes.

Classes supported only in the .NET Compact Framework

When writing device applications, you should know about some classes that are supported only in .NET Compact Framework. You use the Infrared Data Association (IrDA) classes to establish connections to a server and provide infrared port information. SQL Server CE classes enable you to work with a local SQL Server CE database. You will work with SQL Server CE classes in Module 5, Synchronization with SQL Server CE, in Course 2556A, Developing Mobile Applications Using the Microsoft .NET Compact Framework.

For more information about support for individual classes in the .NET Compact Framework, see the .NET Compact Framework SDK documentation.

Key Differences from the .NET Framework

Key Differences from the .NET Framework

Feature	Compact Framework
Common language runtime	Approximately 12 percent the size of the full .NET Framework runtime.
COM Interop	Not supported. Use platform invoke (P/Invoke) to access native DLL functions that then call COM objects.
Data	Provides a subset implementation of ADO.NET and includes the SQL Server CE .NET data provider.
ASP.NET	Not supported. Use ASP.NET Mobile Web Controls to develop Web pages for mobile devices.
XML	Due to size considerations, no support for XML schema validation or XPath queries on XML documents.

Introduction

This module frequently emphasizes the fact that both versions of the .NET Framework share much in common. This topic covers some of the key differences, although the list is by no means comprehensive. For more information about the differences between the Frameworks, see the .NET Compact Framework SDK documentation.

COM Interop

The .NET Compact Framework provides limited support for interoperating with native APIs. You can access functions in native DLLs, but you cannot create COM objects in managed code (COM interop). Also, you cannot call an ActiveX control from managed code.

To call an unmanaged DLL, you must use Platform Invocation Services (PInvoke) by using the **Declare** statement in Visual Basic or the **DLLImport** attribute in Microsoft Visual C#. When you use PInvoke to call unmanaged code from managed code, you can then access native APIs through standard DLL entry points.

Data

You can extend SQL Server to devices through SQL Server CE. Smart Device Extensions adds the SQL Server CE database engine, the **System.Data.SqlServerCe** namespace, and the SQL Server CE .NET data provider to the .NET Compact Framework. You can use SQL Server CE on devices running Windows CE.NET, Pocket PC 2000, or Pocket PC 2002. SQL Server CE supports connections to databases in SQL Server version 7.0 and later.

ASP.NET

The .NET Compact Framework enables you to build rich client applications as opposed to browser-enabled applications. To develop Web pages for mobile devices, you can use the Mobile Internet Toolkit. If you already know ASP.NET and Web Forms, you can easily build mobile Web applications by using mobile Web Forms controls.

XML support

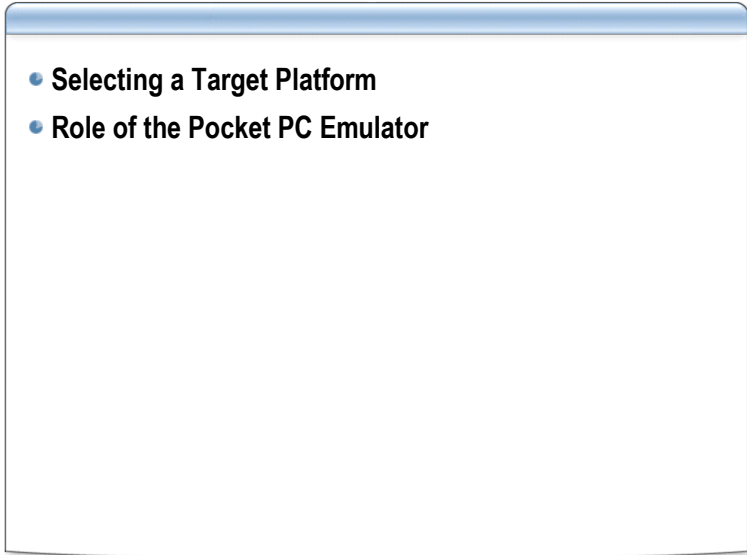
The .NET Compact Framework provides basic XML functionality including the XML Document Object Model (DOM). To ensure that the .NET Compact Framework remains lightweight, the following XML components are not supported:

- XML schema validation
- The **XmlDataDocument** class
- xpath queries, including node selecting methods that take xpath parameters
- Extensible Stylesheet Language Transformation (XSLT)

Module 6, “Designing Applications for a Mobile Environment” ” in course 2556, *Developing Mobile Applications Using the Microsoft .NET Compact Framework*, also lists differences between the Frameworks. However, those differences are specifically related to user interface considerations.

Lesson: Introduction to Smart Device Extensions

Lesson: Introduction to Smart Device Extensions

- 
- Selecting a Target Platform
 - Role of the Pocket PC Emulator

Introduction

The Smart Device Extensions (SDE) for Visual Studio .NET enables you to design, debug, and deploy applications for today's most capable smart devices. As a C# or Visual Basic programmer, you can now create applications for the Pocket PC and other devices running Windows CE .NET.

This lesson introduces the SDE, explains how to select a target platform, and explains how to configure the Pocket PC Emulator.

Lesson objectives

After completing this lesson, you will be able to:

- Explain the role of SDE in the Visual Studio development environment.
- Create a Smart Device Application project and select a target platform.
- Configure the emulator for the target platform.

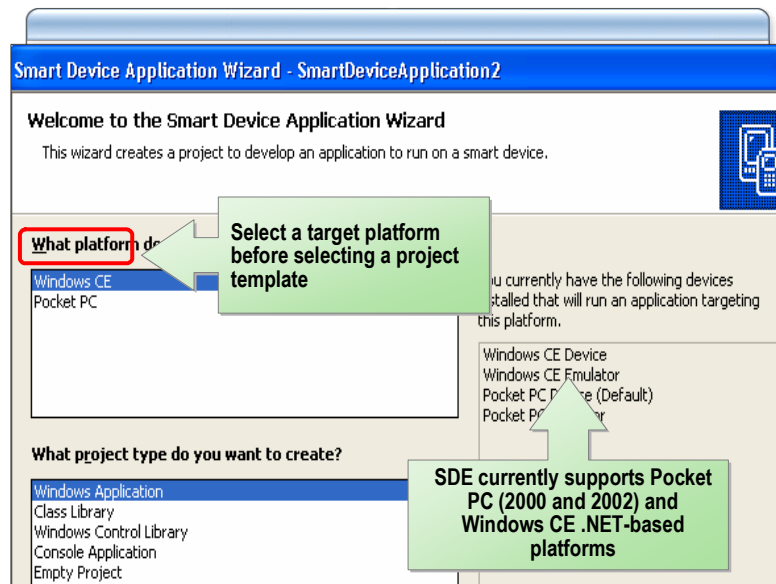
Lesson agenda

This lesson includes the following topics and activities:

- Selecting a Target Platform
- Role of the Pocket PC Emulator
- Demonstration: The Pocket PC Emulator
- Practice: Configuring the Pocket PC Emulator

Selecting a Target Platform

Selecting a Target Platform

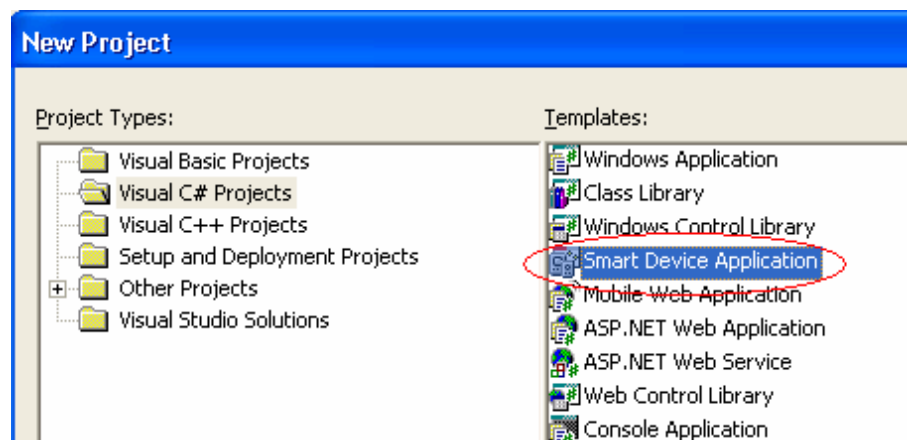


Introduction

When you create a device project, you must first choose a target platform, before you can choose the template for your application (for example, Windows Application). The initial release of SDE ships with support for the following platforms: Windows CE .NET, Pocket PC, and Pocket PC 2002.

Creating a device project

To start a new device project, in Visual Studio, open the **New Project** dialog box and choose a project type and template. The Smart Device Application template is available for Visual C# and Visual Basic projects only.



Definition of a platform

A *platform* in Smart Device Extensions (SDE) is a set of .NET Compact Framework programming functionality that is designed for a group of related devices. Platform class libraries are grouped into *profiles*. A profile is a coherent group of class libraries that is designed for a particular device family, and that has been developed and tested to work well as a unit.

You can change the current target device to any other device supported by the platform of the current project. For example, you might want to change the target from the emulator to the physical device.

Note When you choose a target platform in the Smart Device Application Wizard, the platform is set for that project and cannot be changed. To change the target platform, you must create a new project.

Role of the Pocket PC Emulator

Role of the Pocket PC Emulator

- Provides a virtual computer running on the desktop computer
- Duplicates hardware that runs Microsoft Windows CE on an Intel x86-based PC
- Powered by Windows CE operating system and Pocket PC components
- Guarantees high level of fidelity between actual Pocket PC device and device emulation environment

Introduction

Smart Device Extensions also includes several high fidelity, onscreen device emulators. Emulators are provided for today's Pocket PC, Pocket PC 2002, and Windows CE.NET devices. By using emulators, you can create and test applications without having any devices present.

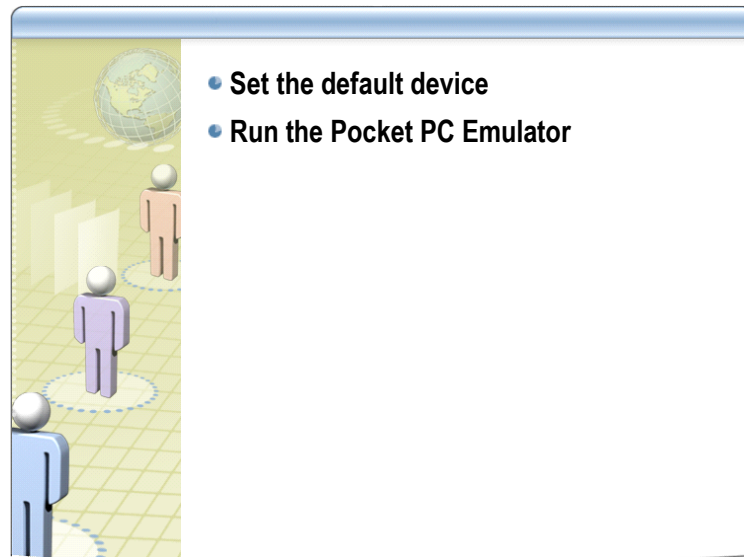
Pocket PC emulation

The Microsoft Windows Powered Pocket PC 2002 SDK includes a new emulation environment. This environment provides a virtual computer running Pocket PC 2002 software compiled for the Intel x86 processor. The virtual computer duplicates hardware that runs Microsoft Windows CE on an x86-based computer.

Previous Windows CE emulators relied on special emulator compilers that passed instructions to the underlying Microsoft Windows NT® operating system. This led to occasional dramatic differences in appearance and function between the emulator and a Pocket PC device. Because the new emulator is powered by the Windows CE operating system and by Pocket PC components, a much higher level of fidelity exists between an actual Pocket PC device and the device emulation environment.

Demonstration: The Pocket PC Emulator

Demonstration: The Pocket PC Emulator



Introduction

This instructor-led demonstration will show you how to set the Pocket PC Emulator as the default device for the Pocket PC platform. You will see how the functionality of the emulator simulates the functionality of a Pocket PC device. You will also see how to set various emulator configuration settings.

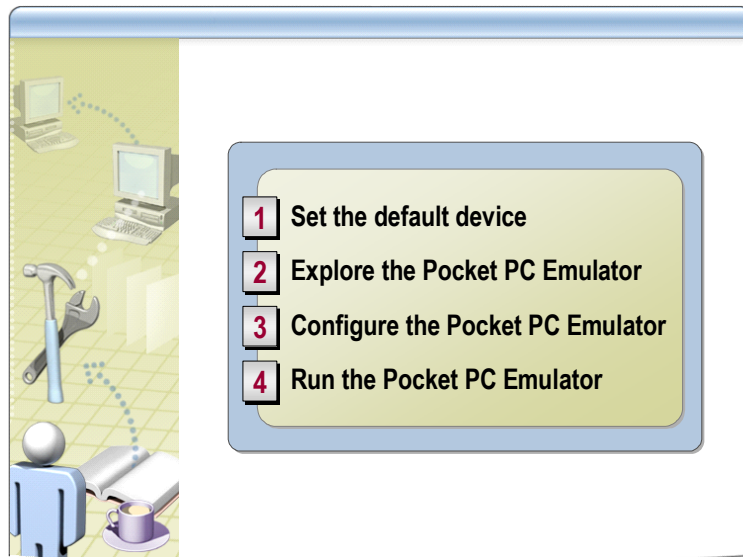
Demonstration

The instructor will:

1. Open Microsoft Visual Studio .NET and create a new Smart Device Application project.
2. Show the configuration options under **Devices** in the **Options** dialog box.
3. Save a copy of the Pocket PC Emulator.
4. Connect to the saved copy of Pocket PC Emulator and show its features and functionality.
5. Configure emulator settings for the saved copy of the Pocket PC Emulator.
6. Run the Emulator with the new settings.

Practice: Configuring the Pocket PC Emulator

Practice: Using the Pocket PC Emulator



Introduction

In this practice, you will set the Pocket PC Emulator as the default device for the Pocket PC platform. You will then explore the functionality of the emulator and note how it simulates the functionality of a Pocket PC device. You will also configure and test various emulator configuration settings.

Practice

► Set the default device for the platform

1. Start Visual Studio .NET and create a new project. The project can be either Visual C# or Visual Basic. Choose **Smart Device Application** as the template.
2. In the Smart Device Application Wizard, choose **Pocket PC** as the target platform and **Windows Application** as the project type. Click **OK**.
3. Note the **Device** toolbar. In the **Deployment Device** list, Pocket PC is set as the default.
4. On the **Tools** menu, click **Options**. In the **Device Tools** folder, click **Devices**.

This displays the device options in the **Options** dialog box. You can also open this dialog box by clicking the **Device Options** button on the **Device** toolbar.

5. Note the platforms that are available in the **Show devices for platform** list. Smart Device Extensions for Visual Studio .NET currently supports Pocket PC 2000, Pocket PC 2002, and any Windows CE .NET-based devices.

6. In the **Devices** box, click **Pocket PC 2002 Emulator**, and then click **Save As**. In the **Save Device As** dialog box, delete any existing text, type **My Pocket PC Emulator** and then click **OK**.

You can save any device or device emulator under a new name. This lets you use custom device or device emulator settings for distinct projects. Notice also that the **Delete** button is now enabled. You can delete only a custom copy of device or device emulator settings.

7. On the **Debug** menu, click **Start** (or use the keyboard shortcut which is **F5**).
8. In the **Deploy *projectname*** dialog box, click **My Pocket PC Emulator**, click **Set As Default**, and then click **Cancel**.

► Explore the Pocket PC Emulator

1. To start the Pocket PC Emulator click **Connect to Device** on the **Device** toolbar.

Notice that the Visual Studio .NET status bar shows the progress of the connection operation as the emulator starts.

2. The emulator may take several moments to start, and then the Welcome screen appears. After the Today screen of the emulator appears, click **Tap here to set owner information**.

3. On the **Owner Information** tab, in the **Name** box, type your name. Notice that the emulator supports a standard keyboard and mouse. Click **OK**

Your name is now displayed as the owner on the main screen.

4. Click **Start** and note the menu items available.

Most of the same menu items that appear on the emulator menu appear on the **Start** menu of a Pocket PC device. Notice that you can establish a connection between the emulator and the desktop computer through ActiveSync.

5. Examine some of the applications in the emulator.

The Pocket PC Emulator has the same functionality as a Pocket PC device. You can perform tasks such as:

- a. Creating files by using Pocket Word and Pocket Excel.
- b. Configuring personal, system, and connection settings.
- c. Sending and reading email, setting tasks and appointments, and creating contacts.

You can even use MSN™ Messenger Service.

6. On the **Emulator** menu, click **Shut Down**.
7. In the **Shut Down** dialog box, in the list, click **Turn off Emulator**, and then click **OK**.

The emulator closes without saving any changes that you made during this practice.

► **Configure emulator settings**

1. On the **Tools** menu, click **Options**, and then in the **Device Tools** folder, click **Devices**.
2. Under **Devices**, click **My Pocket PC Emulator**, and then click **Configure**.
3. In the **Configure Emulator Settings** dialog box, on the **Display** tab, make the following settings.
 - a. Screen width – 480 pixels
 - b. Screen height – 640 pixels
 - c. Color depth – 32

4. Click the **Hardware** tab and note the list items for serial and parallel ports. For this practice, you will leave the default settings as they are.

The emulator provides a parallel port that you can map directly to an LPT port on your development computer. The emulator also provides two serial ports, or virtual COM ports, that you can map to communications (COM) ports on your development computer.

5. Click the **System** tab, and in the **Host key** list, click **Left Control**.

You can also set the amount of memory (in megabytes) that you want to make available to the emulator. The memory size must be a positive integer between 32 and 256. For this practice, you will leave the memory size at 32 megabytes (the default).

6. Click **OK** to close the **Emulator Settings** dialog box, and then click **OK** to close the **Options** dialog box.

► **Test emulator settings**

1. On the **Device** toolbar, click **Connect to Device** to start the emulator.
2. Notice the following changes to the emulator as a result of increased screen resolution:
 - a. The emulator window is larger.
 - b. The colors have changed.
3. Click **Emulator**. Notice that each menu item has an associated shortcut key combination. Recall that in the preceding procedure, you set the Host key to be the left CTRL key.

4. Press CTRL+P to pause the emulator. Press CTRL+P again to resume the emulator.

Clicking **Soft Reset** on the **Emulator** menu simulates pressing the reset button on the Pocket PC device, whereas clicking **Hard Reset** simulates a cold boot of the device.

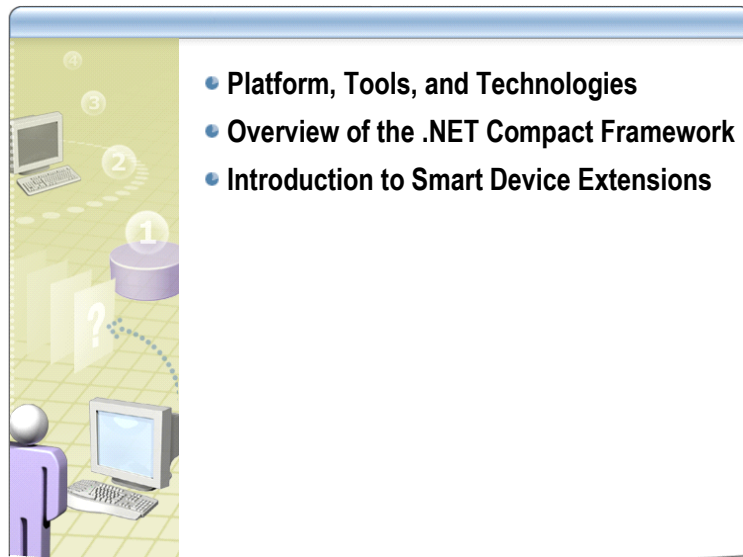
5. Click **Help** to find out more information about basic support for emulated hardware.
6. To close the emulator, press CTRL+F4. In the **Shut Down** list, click **Turn off Emulator**, and then click **OK**.

Important In this practice, you should always close the emulator without

saving the current emulator state. However, if you inadvertently save the emulator state, you can delete My Pocket PC Emulator and reset the default device.

Review

Review



1. What development tool is best suited for a simple application that must target as many device types possible, must be connected most of the time, but has no need to download information to the device?

Use Microsoft Mobile Internet Toolkit to render the application on the devices.

2. What is the best way to build a Pocket PC device application with offline capability that must be delivered from a Web service?

Use Smart Device Extensions for Visual Studio .NET to build a .NET Compact Framework application that enables a Pocket PC device to access the Web service.

3. What component must the computer running IIS have to use the developer edition of SQL Server CE that is included with Smart Device Extensions?

Microsoft SQL Server CE 2.0 Server Tools

4. How do you call native APIs from the .NET Compact Framework?

Use platform invoke (PInvoke) by using the Declare statement in Visual Basic or theDllImport attribute in Visual C#. You can then access native APIs through standard DLL entry points.

5. What two sets of classes are specific to the .NET Compact Framework?

Infrared Data Association (IrDA) classes – used to establish connections to a server and provide infrared port information.

SQL Server CE classes – used to enable working with a local SQL Server CE database.

6. What target platforms does the current release of Smart Device for Extensions support?

Windows CE .NET, Pocket PC 2000, and Pocket PC 2002.