บทที่ 3

ไมโครซอฟต์วิชวลเบสิก

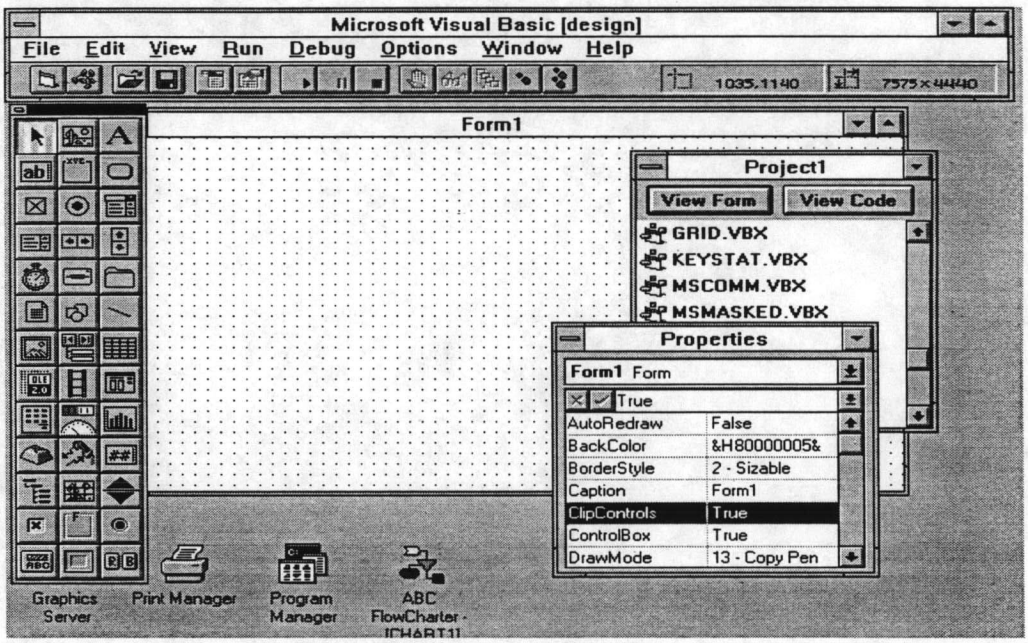
ไมโครซอฟต์วิชวลเบสิก เป็นภาษาทางคอมพิวเตอร์ที่ช่วยในการเขียนโปรแกรมได้รวดเร็วและง่ายที่สุดภาษาหนึ่งสำหรับโปรแกรมประยุกต์ภายใต้ระบบปฏิบัติการวินโดวส์ อีกทั้งสามารถสร้างจอภาพส่วนติดต่อกับผู้ใช้ ที่ดึงดูดความสนใจอันเป็นผลจากการใช้ประโยชน์อย่างเต็มที่จาก Graphical user interface (GUI) โดยจัดเตรียมอุปกรณ์ในการพัฒนาส่วนติดต่อกับผู้ใช้ในลักษณะที่เป็นรูปภาพรูปแบบต่างๆ โปรแกรมเมอร์สามารถสร้างส่วนติดต่อกับผู้ใช้นี้ โดยการวาดอ็อบเจ็ค (object) ที่เป็นส่วนติดต่อในลักษณะที่เป็นรูปภาพ และกำหนดคุณสมบัติ (property) ต่างๆ ของอ็อบเจ็คเหล่านี้ เพื่อกำหนดภาพลักษณ์และพฤติกรรมการแสดงออกทางจอภาพ จากนั้นจึงเขียนส่วนโต้ตอบกับผู้ใช้ โดยการเขียนโปรแกรมเพื่อโต้ตอบกับเหตุการณ์ (event) ที่เกิดขึ้นในระหว่างการติดต่อ

การใช้ไมโครซอฟต์วิชวลเบสิก ยังทำให้โปรแกรมเมอร์สามารถสร้างโปรแกรมประยุกต์ซึ่งใช้ประโยชน์จากคุณสมบัติ (feature) เด่นของไมโครซอฟต์วินโดวส์ ได้แก่ การมีส่วนติดต่อกับผู้ใช้หลายจอภาพพร้อมกันในเวลาเดียวกัน (Multiple-document interface หรือ MDI), การเชื่อมและผนึกรวมก้อนของอ็อบเจ็ค (object linking and embeddedding หรือ OLE), การแลกเปลี่ยนข้อมูลแบบพลวัต (dynamic data exchange หรือ DDE) และอีกมากมาย

นอกจากนี้ไมโครซอฟต์วิชวลเบสิกยังสามารถเพิ่มจำนวนฟังก์ชันเพื่อขยายการใช้งานโดยการเพิ่มส่วนควบคุมประดิษฐ์ (custom control) ไว้ในโปรแกรมประยุกต์ ซึ่งอาจจะโดยการเรียกใช้กระบวนการต่างๆ ในคลังคำสั่งการเชื่อมแบบพลวัต (dynamic-link libraries) หรือ โดยการเรียกใช้กระบวนการต่างๆ ในวินโดวส์ API ก็ได้

ขั้นตอนการสร้างโปรแกรมประยุกต์ด้วยวิชวลเบสิก

ขั้นตอนแรกของการสร้างโปรแกรมประยุกต์ด้วยวิชวลเบสิก คือการสร้างส่วนติดต่อกับผู้ใช้ ซึ่งได้แก่แบบฟอร์ม (form), ส่วนควบคุม (control) และอ็อบเจ็คอื่นๆ ที่ผู้ใช้สามารถเห็นและใช้งาน จากนั้นจึงกำหนดคุณสมบัติของแบบฟอร์ม และส่วนควบคุมเพื่อกำหนดค่าต่างๆ เช่นหัวเรื่องหรือคำอธิบายประกอบภาพ (caption), สีและขนาด ขั้นตอนสุดท้ายคือเขียนส่วนโปรแกรมเพื่อทำให้ส่วนติดต่อนี้ทำงาน (อ่านรายละเอียดขั้นตอนการเขียนโปรแกรมประยุกต์ภายใต้วิชวลเบสิกได้จากหนังสือ Programmer's Guide ในชุดหนังสือ Microsoft Visual Basic รุ่นที่ 3.0 หน้า 18) รูปที่ 3.1 แสดงจอภาพสภาพแวดล้อมสำหรับการเขียนโปรแกรมภาษาวิชวลเบสิก ซึ่งประกอบไปด้วย แถวเครื่องมือ (toolbar), กล่องเครื่องมือ (toolbox), แบบฟอร์ม (form), แถวเมนู (menu bar), หน้าต่างของโปรเจ็ค (Project window) และหน้าต่างของคุณสมบัติ (Properties window)



รูปที่ 3.1 แสดงจอภาพสภาพแวดล้อมสำหรับการเขียนโปรแกรมวิชวลเบสิก

โปรแกรมเมอร์ใช้ส่วนควบคุมเพื่อรับข้อมูลนำเข้าจากผู้ใช้และแสดงผลลัพธ์ทางจอภาพ ส่วนควบคุมบางส่วน สามารถนำมาใช้ในโปรแกรมประยุกต์ได้ ได้แก่ กล่องตัวอักษร (text box), ปุ่มคำสั่ง (command button) และ กล่องรายการ (list box) ส่วนควบคุมแต่ละประเภท มีชุดของคุณสมบัติและเหตุการณ์ของตัวเอง รูปที่ 3.1 ทางด้านซ้ายมือ แสดงกล่องเครื่องมือที่มีใช้ใน

วิชวลเบสิก โดยกล่องเครื่องมือนี้จะเก็บเครื่องมือที่ใช้ในการเขียนส่วนควบคุมต่างๆ ลงบนแบบฟอร์ม เครื่องมือแต่ละตัวใช้แทนหนึ่งส่วนควบคุม

การใช้ฟังก์ชันส่วนขยายในวิชวลเบสิก

จากที่ได้กล่าวในข้างต้นว่าไมโครซอฟต์วิชวลเบสิก สามารถเพิ่มจำนวนฟังก์ชันเพื่อขยายการใช้งานโดยการเพิ่มส่วนควบคุมประดิษฐ์ไว้ในโปรแกรมประยุกต์, โดยการเรียกใช้กระบวนการต่างๆ ในคลังคำสั่งการเชื่อมแบบพลวัต (DLL) หรือโดยการเรียกใช้กระบวนการต่างๆในวินโดวส์ API การเรียกใช้ฟังก์ชันจากทั้ง 3 ส่วนนี้ มีข้อแตกต่างกันอยู่บ้างดังจะกล่าวต่อไปนี้

1. ส่วนควบคุมประดิษฐ์ (Custom control)

ส่วนควบคุมประดิษฐ์จะรวบรวมฟังก์ชันเก็บไว้ในแฟ้มข้อมูลต่างๆ แยกตามลักษณะการใช้งานโดยมีนามสกุล (extension) ของแฟ้มข้อมูลเป็น .VBX เช่น MSCOMM.VBX และ MSMAPI.VBX เป็นต้น โดย MSCOMM.VBX ถูกใช้ในงานเกี่ยวกับการสื่อสารผ่านโมเด็ม และ MSMAPI.VBX ใช้ในงานเกี่ยวกับการรับ-ส่งข้อมูลข่าวสารไปรษณีย์อิเลคทรอนิกส์

ฟังก์ชันภายใต้แฟ้มข้อมูลนามสกุล .VBX แท้ที่จริงก็คือตัวแทนฟังก์ชันในคลังคำสั่งการเชื่อมแบบพลวัต หรือ วินโดวส์ API ซึ่งบริษัทผู้ขายหรือตัวแทนบริษัทไมโครซอฟต์ได้พัฒนาขึ้นเพื่อให้โปรแกรมเมอร์ภาษาวิชวลเบสิกสามารถเรียกใช้ได้โดยสะดวก การเรียกใช้ฟังก์ชันใน .VBX เรียกใช้ได้ง่ายกว่าการเรียกใช้ฟังก์ชันในคลังคำสั่งการเชื่อมแบบพลวัต หรือ วินโดวส์ API เนื่องจากไม่ต้องกำหนดอาร์กิวเมนต์ยุ่งยาก

2. คลังคำสั่งการเชื่อมแบบพลวัต (Dynamic-link libraries) (DLL)

คลังคำสั่งการเชื่อมแบบพลวัต เป็นคุณสมบัติสำคัญของไมโครซอฟต์วินโดวส์ โดยถูกเขียนขึ้นมาเพื่อใช้เฉพาะงาน และมีนามสกุลของแฟ้มข้อมูลเป็น .DLL ส่วนในการเรียกใช้งานโปรแกรมประยุกต์จะขอเชื่อมโยง (link) และเรียกใช้ในขณะทำงานตามชุดคำสั่ง (run time) โปรแกรมประยุกต์หลายโปรแกรม สามารถแบ่งปันการใช้งาน DLL ในเวลาเดียวกันได้ ขณะเดียวกันตัวไมโครซอฟต์วินโดวส์เอง ก็ประกอบขึ้นจากหลาย DLL ซึ่งรวบรวมกระบวนการต่างๆ ที่โปรแกรมประยุกต์ทุกโปรแกรม สามารถเรียกใช้เพื่อทำกิจกรรมต่างๆ เช่น การแสดงจอภาพและ

รูปภาพ, การจัดการหน่วยความจำและอื่นๆ (กระบวนการเหล่านี้บางครั้งก็เรียกว่าเป็น วินโดวส์ API)

โปรแกรมประยุกต์ภาษาวิชวลเบสิก สามารถเรียกใช้กระบวนการใน DLL เหล่านี้เพื่อเรียกการทำงานพิเศษที่โปรแกรมเมอร์ไม่สามารถเรียกใช้โดยตรงได้ในวิชวลเบสิก โปรแกรมเมอร์สามารถเรียกกระบวนการใน DLL อื่นๆ ที่มีอยู่ในระบบเช่นกัน

3. วินโดวส์ API

การเขียนโปรแกรมโปรแกรมประยุกต์บนวินโดวส์ โปรแกรมเมอร์สามารถเขียนโปรแกรมเพื่ออ่านข้อมูลบางส่วนโดยตรงจากวินโดวส์ โดยการใช้วินโดวส์ API (Windows application programming interface) ซึ่งเป็นชุดของฟังก์ชันมากกว่า 700 ฟังก์ชัน ฟังก์ชันเหล่านี้ส่วนใหญ่ จะเก็บไว้ในแฟ้มข้อมูล KERNEL.EXE, USER.EXE และ GDI.EXE เป็นหลัก และสามารถเรียกใช้ได้ในทุกโปรแกรมประยุกต์ ตัวอย่างเช่น หากต้องการทราบข้อมูลบางส่วนจากแฟ้มข้อมูล INI ซึ่งโปรแกรมบนวินโดวส์ส่วนใหญ่รวมทั้งตัววินโดวส์เองจะมีแฟ้มข้อมูล INI เพื่อเก็บรายละเอียดของโปรแกรม เช่น แฟ้มข้อมูล WIN.INI และ SYSTEM.INI จะเก็บข้อมูลสำหรับการเริ่มต้นทำงานของวินโดวส์ และ ข้อมูลเกี่ยวกับระบบ โปรแกรมเมอร์สามารถใช้โปรแกรม SYSEDIT.EXE จากไดเร็กทอรีของวินโดวส์ เพื่ออ่านและแก้ไขแฟ้มข้อมูล INI นั้นได้ เนื่องจากความต้องการอ่านข้อมูลจากแฟ้มข้อมูล .INI มีอยู่อย่างสม่ำเสมอ วินโดวส์จึงได้จัดให้มีฟังก์ชันเพื่อโปรแกรมประยุกต์สามารถเรียกใช้ในการอ่าน และ แก้ไขแฟ้มข้อมูลเหล่านี้ได้โดยสะดวก ฟังก์ชันที่ว่านี้ก็คือส่วนหนึ่งของวินโดวส์ API นั่นเอง

ตัวอย่างเปรียบเทียบการเรียกใช้ฟังก์ชัน

ตัวอย่างต่อไปนี้ เป็นตัวอย่างในการลงชื่อเข้าสู่ระบบ ซึ่งจะแยกเปรียบเทียบเป็น 2 กรณี คือ เมื่อเรียกใช้ด้วยฟังก์ชันจากคลังคำสั่งสำหรับการเชื่อมแบบพลวัต ซึ่งในที่นี้คือ MAPI.DLL และเรียกใช้ด้วยฟังก์ชันจากส่วนควบคุมประดิษฐ์ที่ชื่อว่า MSMAPI.VBX

เมื่อเรียกใช้ด้วยฟังก์ชันจากคลังคำสั่งการเชื่อมแบบพลวัต โปรแกรมเมอร์สามารถเรียกใช้ด้วยคำสั่ง MAPILogon ผลของการเรียกใช้จะให้ค่า session handle ภายใต้ตัวแปร

MAPISession ซึ่งจะต้องนำไปใช้ในฟังก์ชันอื่นต่อไป หากต้องการทำงานกับไปรษณีย์อิเลคทรอนิกส์ รูปที่ 3.2 แสดงตัวอย่างการเรียกใช้ฟังก์ชันจากคลังคำสั่งการเชื่อมแบบพลวัต

```
Declare Function MAPILogon Lib "MAPI.DLL" (ByVal UIParam&, ByVal User$, ByVal
Password$, ByVal Flags&, ByVal Reserved&, Session&) As Long

MAPISession = 0
e& = MAPILogon (0, "", "", MAPI_LOGON_UI, 0, MAPISession)
If (e& <> success_success) Then
  MsgBox "MAPILogon error=" + Str$(e&)
End If
```

รูปที่ 3.2 แสดงตัวอย่างการเรียกใช้ฟังก์ชันจากคลังคำสั่งการเชื่อมแบบพลวัต

เมื่อจะเรียกใช้ด้วยฟังก์ชันจาก MSMAPI.VBX จะเรียกใช้ด้วยการกำหนดคุณสมบัติของ ACTION เป็น SESSION_SIGNON โดยค่า SESSION_SIGNON มีค่าคงที่เป็น 1

```
VBAPMT.MapiSess.Action = SESSION_SIGNON
If err <> 0 then
  MsgBox "Logon Failure: " + Error$
End If
```

รูปที่ 3.3 แสดงตัวอย่างการเรียกใช้ฟังก์ชันจากส่วนควบคุมประดิษฐ์

จากตัวอย่างดังรูปที่ 3.3 เมื่อเรียกใช้ด้วยการกำหนดคุณสมบัติของ ACTION จะได้ค่าควบคุม session โดยถูกเก็บไว้ในคุณสมบัติ VBAPMT.MapiSess.SessionID ซึ่งจะต้องนำไปใช้ในฟังก์ชันอื่นเช่นกัน หากต้องการทำงานกับไปรษณีย์อิเลคทรอนิกส์ต่อไป

จากตัวอย่าง จะพบว่าการเรียกใช้ฟังก์ชันจากส่วนควบคุมประดิษฐ์ สามารถเรียกใช้ได้ง่ายกว่า อย่างไรก็ตามในงานบางงาน อาจจะไม่สามารถหลีกเลี่ยงการใช้ฟังก์ชันจากคลังคำสั่งการเชื่อมแบบพลวัต หรือ ฟังก์ชันจากวินโดวส์ API ได้ ปัญหาหนึ่งที่จะต้องระวังในการเรียกใช้ฟังก์ชันเหล่านี้คือ ถ้ากำหนดค่าอาร์กิวเมนต์ไม่ถูกต้องด้วยเหตุผลใดก็ตาม ระบบอาจจะถูกล็อคและต้องแก้ไขด้วยการบูทเครื่องใหม่ จึงแนะนำให้ทำการเซฟ (save) โปรแกรมอย่างสม่ำเสมอในขณะทดสอบโปรแกรม เพื่อลดการเสียเวลาในการที่จะต้องพิมพ์ใหม่ในกรณีที่เครื่องหยุดการทำงานกระทันหัน (hang)

สรุปขั้นตอนการเรียกใช้ฟังก์ชันจากคลังคำสั่งการเชื่อมแบบพลวัตหรือฟังก์ชันจากวินโดว์ API

ขั้นตอนการเรียกใช้ฟังก์ชันจากคลังคำสั่งการเชื่อมแบบพลวัตหรือฟังก์ชันจากวินโดวส์ API สามารถสรุปการเรียกใช้ได้ด้วยขั้นตอนทั่วไปต่อไปนี้

1. ให้คำนิยามฟังก์ชันด้วยคำสั่ง DECLARE (เนื่องจากวิชวลเบสิกไม่รู้จักฟังก์ชันเหล่านี้)
2. กำหนดเนื้อที่สำหรับตัวแปรด้วยคำสั่ง Global หรือ Dim เป็นต้น (ฟังก์ชันต้องการเนื้อที่สำหรับคำตอบ)
3. เรียกใช้ฟังก์ชัน

```
1.  Declare Funtion GetWindowsDirectory Lib "KERNEL.EXE" (ByVal lpBuffer As String,
    ByVal nSize As Integer) As Integer
2.  Dim WinPath As String * 255
3.  Worked = GetWindowsDirectory(WinPath, Len(WinPath))
```

รูปที่ 3.4 แสดงตัวอย่างการเรียกใช้ฟังก์ชันจากวินโดวส์ API

รูปที่ 3.4 แสดงตัวอย่างการเรียกใช้ฟังก์ชันจากวินโดวส์ API ตามลำดับขั้นตอน 1, 2 และ 3 โดยฟังก์ชันเหล่านี้ ใช้ในการนิยามฟังก์ชันเพื่อใช้ในการค้นหาที่ตั้งของไดเร็กทอรีของวินโดวส์ ซึ่งส่วนใหญ่จะอยู่บนไดรว์ C เช่น C:\WINDOWS แต่อาจจะมีบ้างที่แตกต่างไปจากนี้ ฟังก์ชันดังกล่าวอยู่ในแฟ้มข้อมูลชื่อ KERNEL.EXE โดยตัวฟังก์ชันมีชื่อว่า GetWindowsDirectory เมื่อเรียกใช้ ฟังก์ชันจะให้ค่าเส้นทาง (Path) ของไดเร็กทอรีของวินโดวส์ในรูปของตัวอักษร (string) ในที่นี้คือตัวแปรชื่อ WinPath

ลำดับขั้นการเรียกใช้คำสั่งคือ ขั้นแรกใช้คำสั่ง Declare เพื่อบอกให้วิชวลเบสิกรู้ว่าฟังก์ชันนั้นอยู่ในแฟ้มข้อมูลอะไร ตลอดจนค่าต่างๆ ที่ต้องส่งไป จากนั้นให้กำหนดเนื้อที่ตัวแปรสำหรับการส่งค่ากลับ ในที่นี้คือ WinPath ซึ่งเป็นตัวอักษรความยาว 255 ตัวอักษร ในขั้นตอนสุดท้ายคือการเรียกใช้งานด้วยคำสั่ง Worked = ถ้าฟังก์ชันทำงานสำเร็จ ก็จะให้ค่า Worked เป็นศูนย์ แต่ถ้ามีปัญหาเกิดขึ้นจะให้ค่าเป็นค่าเลขที่ของข้อผิดพลาด (error code) เพื่อให้โปรแกรมเมอร์ตรวจสอบโปรแกรมต่อไป