EMBARCADERO
TECHNOLOGIES®

2011 Delphi 技術講座:

將舊版 Delphi 專案升級為Unicode相容版本

張子仁
元智大學資傳系兼任講師
易繁科技技術總監

講座大綱

- 基本概念介紹
  - Unicode的重要性
  - Delphi 與 Unicode 的關係
- 將舊版專案改造為 Unicode 相容的版本
  - Delphi 7-2007 專案中與 Unicode 有關的部分
  - Delphi 中常見資料處理方式需額外處理的部分
  - 改造專案四大分類
- WorkShop
- Q&A

EMBARCADERO
TECHNOLOGIES®

基本概念介紹

Unicode的重要性

- 以單一編碼涵蓋了超過十萬個全世界現存語言的文字
- 已經成為全球通用的標準
- 簡化多國語文版本程式製作上的程序
- 從 Windows 2000 起, 成為 Windows 作業系統內部顯示的文字標準

Delphi 與 Unicode 的關係 (I)

- 自 Delphi 2009 起, 全架構均為 Unicode 相容版本: 自 IDE 到 VCL 架構都是
- Delphi 2009 以前的版本: IDE 是 ANSI 相容, VCL 則可以透過 3rd Party 元件提供 Unicode 功能, 可惜仍不完整。 (tntUnicodeControls, 後更名為 tntWares)
- 以 ANSI 版的 Delphi (1-2007)開發時，可透過 Windows API, 將資料轉換為 Unicode. (使用 ANSI 版的第三方元件時可用此法改成 Unicode相容)

Delphi 與 Unicode的關係 (II)

重要資料型別在不同版本的Delphi中的關係

型別名稱	Delphi 2009以前	Delphi 2009以後
AnsiChar	在記憶體中以1 byte空間儲存資料型別	在記憶體中以1 byte空間儲存資料型別
Char	在記憶體中以1 byte空間儲存資料型別	在記憶體中以2 bytes儲存空間的資料型別
WideChar	在記憶體中以2 bytes儲存空間的資料型別	在記憶體中以2 bytes儲存空間的資料型別
String	可以視為Char的動態陣列	可以視為WideChar的動態陣列

將舊版專案改造為 UNICODE 相容的版本

Delphi 7-2007專案中與Unicode有關的部分

- Delphi 專案與程式碼當中與字元編碼相關的部分包含有:
  - 介面上的文字顯示 (DFM中的元件)
  - ShowMessage/MessageDlg 要顯示的文字
  - 檔案/資料庫中的文字資料處理
  - 內部用來處理文字的 Class

Delphi 中常見資料處理方式需額外處理的部分 (I)

- 將 String 當作不定長度陣列，用以處理二進位資料

```
var
   data : AnsiString;
   singleByte : AnsiChar;
begin
   data := LoadFromStream...
   ...
   singleByte := data[10];
```

- 解決方法:
  - 將該變數的型別從String改為AnsiString或者ByteString
  - 單一byte的部分則從Char改為AnsiChar

Delphi 中常見資料處理方式需額外處理的部分 (II)

- 將TStrings/TStringList 作為文字編輯或文字檔案暫存、文字排序、物件排序的Class
  - 載入與儲存部分需修改

```
var
   data : TStringList;
begin
   data := TStringList.LoadFromFile(檔案路徑,TEncoding.ASCII);
   ……
   data.SaveToFile(檔案路徑,TEncoding.ASCII)
```

  - 須加入Encoding的描述: TEncoding.ASCII、TEncoding.UTF8等

Delphi 中常見資料處理方式需額外處理的部分 (III)

- 使用3rd Party 提供的 VCL元件提供程式功能 (I)
  - 有Source code 時
    - 修改文字處理部分、TStrings、二進位處理部分為 Unicode 相容

Delphi 中常見資料處理方式需額外處理的部分 (III)

- 使用3rd Party 提供的 VCL元件提供程式功能 (II)
  - 沒有Source code時
    - 製作繼承該元件的 Unicode 相容版本
    - 將該元件所有原本以String傳遞的參數、屬性在宣告中改為AnsiString，並在繼承該元件的版本中將這些參數、宣告完全修改為String
    - Implementation的部分，則需要接收原始元件的資料、並以MultiByteToWideChar將這些Ansi資料轉換為Unicode
    - 發送資訊的部分，則需要以WideCharToMultiByte把Unicode轉換為Byte array後再行傳遞

改造專案-分類一(I)

- 分類一: 純使用 Delphi VCL，無第三方元件
  - 步驟:
    - 以Unicode 版 Delphi 開啟舊版專案, 將舊版專案另存為 Unicode 版專案.
    - 介面修改: DFM都不用更動，可立即使用.
    - 程式碼修改:
      - 確認原本程式碼裡面輸入的字串常數 (例: '直接寫死的常數')
      - 修改以 string 處理二進位資料的程式部分
      - 修改檔案載入/儲存時的Function (增加 TEncoding 的使用)
      - 確認 Registry 讀寫的資料無誤 (重新編譯後以 debug 模式確認)

改造專案-分類一(II)

- 步驟 (續前頁):
  - 驗證: 重新 Compile, 測試, 驗證以下的點:
    - Registry 讀寫資料是否都正確無誤
    - 對不同國別的檔案名稱、文字檔案內容讀寫是否正確
    - 在不同語系的 Windows 作業系統上執行重新編譯後的執行檔，驗證介面文字顯示是否都正確
  - 非 Unicode 相容的程式
    - 只能讀寫英文跟執行程式的 Windows 系統所使用的語系所用的檔名 (例如: 在繁體中文版的 Windows 作業系統上執行的舊版程式，只能讀寫繁體中文跟英文檔名)
    - 如果Registry的Value 使用中文，則舊版程式在日文或英文版 Windows 系統上執行時，讀寫 Registry 資料會取得錯誤資訊

改造專案-分類二(I)

- 分類二: Delphi VCL，使用 ANSI 版第三方元件
  - 步驟:
    - 以Unicode 版 Delphi 開啟舊版專案, 將舊版專案另存為 Unicode 版專案.
    - 元件修改:
      - 新增元件，繼承原來使用的ANSI版第三方元件
      - 將原來元件中所有型別為 String 的屬性，原始 pas 檔宣告的部分都 改為 AnsiString, 並在新的介面中將同一個屬性 override, 設定其型別為 String.

改造專案-分類二 (II)

- 步驟 (續前頁):
  - 在Implementation的部分，呼叫原始元件的Method，取得回傳值之後，先以MultiByteToWideChar將原始 Method 的回傳值從 AnsiString/AnsiChar 改為 String/WideChar 之後再行回傳
  - 重新 Compile BPL, 同時安裝新舊版本元件
  - 將新版專案中，DFM 裡面使用到舊版元件的元件型別，全部改為 Unicode 版元件的 Class 名稱
    - » DFM檔案從 Delphi 5開始就提供文字模式儲存，因此我們可以用Delphi 的編輯器來編輯它，也可以使用您慣用的文字編輯器來進行尋找與取代的程序

改造專案-分類二(III)

- 步驟 (續前頁):
  - 依照分類一專案的處理方式，將程式碼當中與Unicode不相容的部分修改好
  - 重新編譯
  - 以分類一的測試方式，在不同語系的系統上測試新的執行檔、開啟/儲存不同語系的文字檔案

改造專案-分類三(I)

- 分類三: 混用標準VCL 與 tntUnicodeControls 元件
  - 步驟:
    - 先備份舊版專案的所有檔案
    - 以Unicode 版 Delphi 開啟舊版專案, 將舊版專案另存為 Unicode 版專案.
    - 將DFM檔案裡所有以Ttnt開頭的元件型別，全改成T開頭的型別名稱
      - 例外情況:如果該Ttnt開頭的元件是您自己開發的，請維持該元件的型別名稱，但其中使用到tntUnicodeControls 標準元件的部分，記得都要修改喔

改造專案-分類三(II)

- 步驟 (續前頁):
  - 將所有程式碼當中，使用到tntUnicodeContorl標準元件的部分，都改為使用標準 VCL 元件
    - 請記得tntUnicodeControl元件並沒有完整的開發出所有標準VCL的元件
    - TtntStrings 與 TtntStringList 的載入與儲存檔案的部分，並沒有指定TEncoding的功能，這部分要特別注意
  - 如果專案中有使用到Ansi版的第三方元件，請依照分類二的修改方式，以繼承的方式製作出該元件相容於 Unicode 的版本
    - 新版的BPL需要重新編譯、安裝
    - 新舊版的BPL與元件檔案需要同時安裝在系統中才能正常運作

改造專案-分類三(IV)

- 步驟 (續前頁):
  - 請確認新版 BPL 功能運作無誤之後，再繼續處理專案更新的其他部分
  - 依照分類一與分類二的其餘處理流程，將專案的pas程式碼進行更新，使之能夠相容於 Unicode.
  - 重新編譯、在不同語系的Windows系統上進行測試

改造專案-分類四(I)

- 分類四: 使用Indy 的專案
  - 幾乎無法直接升級
    - Indy 8->Indy 9-> Indy 10, 每一版本都有內部自行定義製作的Class, 回傳的資料型別幾乎都不相同
    - Indy 10.2才開始支援 Unicode
  - 只能以舊有的程式進行改寫
  - 改寫步驟:
    - 同樣的，將所有與String相關的型別都標註出來
    - 從 Indy 10.5.2 版開始，許多型別都已經改為 TIdBytes, 改寫時必須很仔細

改造專案-分類四(II)

- 步驟 (續前頁):
  - 在不同語系的作業系統上進行測試
  - 如果Client與Server都是自行開發，切記一定要重新測試整個 Protocol 的正確性，以避免由於編碼資訊導致 Protocol 無效

WORKSHOP

Q&A