PART 5

網路與安全篇

Q53 我要如何在 ASP.NET 中與公司內其他的 FTP 伺服器連結？

適用範圍： V ASP.NET 2.0 V ASP.NET 3.5

問題

我的公司有一台 FTP 伺服器, 用來存放使用者的一些業務檔案資料, 老闆說要我設計一個 Web 介面來讓公司的人員可以由 Web 介面來下載和上傳他們的檔案, 請問我應該要怎麼設計？

問題說明

FTP Server 和 Web Server 可以說是前端資料服務的二個主要服務, 檔案通常都會由 FTP Server 來管理, 而資料的呈現則來自於 Web Server, 因爲 Web 可以做到較豐富的資料展現, 以及多媒體的應用, 而 FTP 就只能做到檔案管理與傳輸的功能, 但 FTP 簡單, Web 相形之下就顯得複雜多了。

由於要由瀏覽器連接 FTP Server 其實不是一件方便的事, 瀏覽器在連接 FTP Server 時速度會變慢, 而且在一些 FTP 的設定上也無法支援, 大多數的使用者都會用 FTP 的用戶端工具來做, 例如 WS-FTP、CuteFTP、FileZilla 等工具, 不過在使用 Web 之虞又要額外使用 FTP 用戶端工具, 在使用上其實還是有點不太方便, 因此 Web 應用程式與 FTP Server 的連接就會變得有所需求。

FTP 也是 Internet 服務的一種, 其規範列在 RFC 959 中, 定義了數組指令, 讓 FTP Client 可以透過 Port 20/21 來和 FTP Server 連接, 進行檔案管理與傳輸的工作。在.NET 2.0 中, 有一組專門處理 FTP 工作的類別：FtpWebRequest 以及 FtpWebResponse 二個, 並且.NET 將 FTP 會用到的指令包裝在 WebRequestMethod.Ftp 類別中, 其他的操作和 HttpWebRequest/HttpWebResponse 相當類似, 一樣是經由 NetworkStream 來讀 (下載) 寫 (上傳) 資料。

FtpWebRequest 提供了幾個 FTP 的作業方式, 開發人員可以選用適當的方式來連接 FTP Server 並執行所需的工作 (這些都包在 WebRequestMethods.Ftp 類別中, 每一個工作都是靜態成員) 。

FTP 方法	說明
AppendFile	表示 FTP APPE 通訊協定方法，用來將檔案附加至 FTP 伺服器上的現有檔案。
DeleteFile	表示 FTP DELE 通訊協定方法，用來刪除 FTP 伺服器上的檔案。
DownloadFile	表示 FTP RETR 通訊協定方法，用來從 FTP 伺服器下載檔案。
GetDateTimestamp	表示 FTP MDTM 通訊協定方法，用來擷取 FTP 伺服器上檔案的日期時戳。
GetFileSize	表示 FTP SIZE 通訊協定方法，用來擷取 FTP 伺服器上檔案的大小。
ListDirectory	表示 FTP NLIST 通訊協定方法，用來取得 FTP 伺服器上檔案的簡短列表。
ListDirectoryDetails	表示 FTP LIST 通訊協定方法，用來取得 FTP 伺服器上檔案的詳細列表。
MakeDirectory	表示 FTP MKD 通訊協定方法，用來在 FTP 伺服器上建立目錄。
PrintWorkingDirectory	表示 FTP PWD 通訊協定方法，用來顯示目前工作目錄的名稱。
RemoveDirectory	表示移除目錄的 FTP RMD 通訊協定方法。
Rename	表示重新命名目錄的 FTP RENAME 通訊協定方法。
UploadFile	表示將檔案上載至 FTP 伺服器的 FTP STOR 通訊協定方法。
UploadFileWithUniqueName	表示將檔案名稱爲唯一的檔案上載至 FTP 伺服器的 FTP STOU 通訊協定方法。

(本表取自.NET Framework SDK 中, WebRequestMethods.Ftp 類別所有成員列表中的說明)

但若是.NET 1.x 的開發人員, 就只能使用 System.Net.Socket 命名空間中的 TcpClient 類別來實作 FTP 的通訊方法, 並且使用原始的 FTP 指令來處理。

解決方案

利用 FtpWebRequest 與 FtpWebResponse 連接 FTP Server 並執行工作, 例如若想要列舉某一個 FTP Server 根目錄下的子目錄, 可以用這樣的方法：

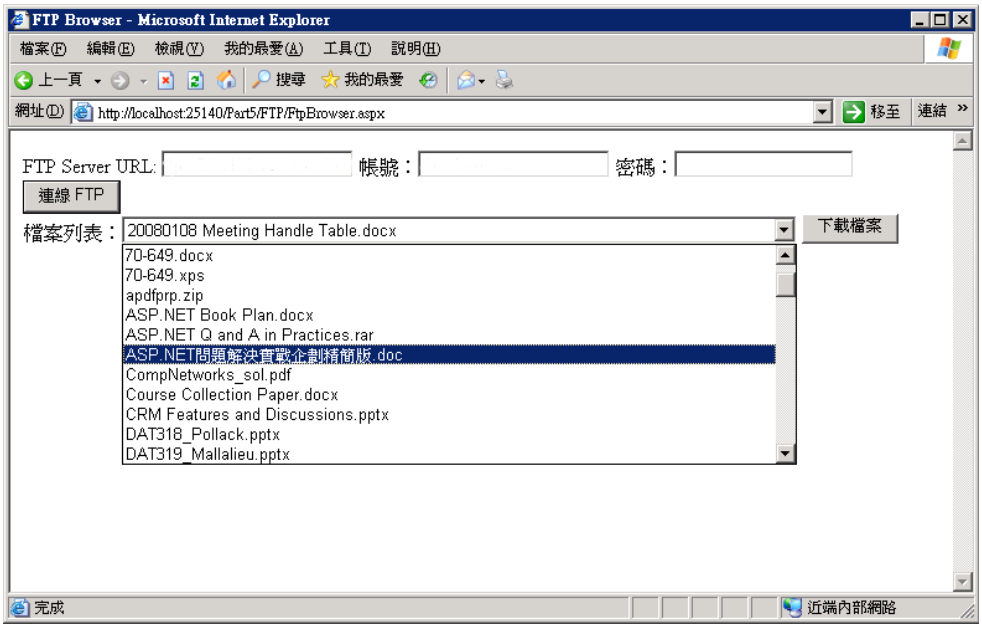
```
FtpWebRequest request = FtpWebRequest.Create(this.T_FtpUrl.Text)
  as FtpWebRequest;
// 設定 FTP 連線屬性。
request.Method = WebRequestMethods.Ftp.ListDirectory;
request.KeepAlive = true;
request.Credentials = new NetworkCredential(this.T_UserName.Text,
  this.T_Password.Text);
request.UsePassive = false;
request.ReadWriteTimeout = 2000000;
request.Timeout = 1200000;
// 儲存帳戶資訊。
Dictionary<string, string> accountInfo =
  new Dictionary<string, string>();
accountInfo.Add("UserName", this.T_UserName.Text);
accountInfo.Add("Password", this.T_Password.Text);
ViewState["FtpAccount"] = accountInfo;
// 下載目錄清單。
FtpWebResponse response = request.GetResponse() as FtpWebResponse;
StreamReader sr = new StreamReader(response.GetResponseStream(),
  System.Text.Encoding.Default);

do
{
  // 讀取檔案清單並加入下拉式方塊中。
  string item = sr.ReadLine();
  if (item.IndexOf('.') >= 0)
```

```
    this.cboFileList.Items.Add(new ListItem(item, item));
}
while (!sr.EndOfStream);

response.Close();
```

本程式是以抓取 FTP 下的檔案並放到檔案列表的下拉式選單中, 其執行結果如下：



若想要由 FTP 下載檔案, 可以用下列的程式碼 (其實只是換一個 FTP Method, 然後把讀寫檔案的方法改變一下。

```
// 建立存取檔案的 FTP URL (例如：ftp://ftp.myftp.org/myfile.zip)
FtpWebRequest request = FtpWebRequest.Create(
  this.T_FtpUrl.Text + "/" +
  this.cboFileList.SelectedValue) as FtpWebRequest;
```

```
// 如果有在頁面中使用 ViewState 來暫存帳戶密碼, 取出帳戶密碼。
Dictionary<string, string> accountInfo =
ViewState["FtpAccount"] as Dictionary<string, string>;
// 設定 FTP 方法, 下載檔案
request.Method = WebRequestMethods.Ftp.DownloadFile;
request.KeepAlive = true;
request.Credentials = new NetworkCredential(
accountInfo["UserName"], accountInfo["Password"]);
request.UsePassive = false;
request.ReadWriteTimeout = 2000000;
request.Timeout = 1200000;

FtpWebResponse response = request.GetResponse() as FtpWebResponse;
// 讀取下載資料流。
BinaryReader reader = new BinaryReader(response.GetResponseStream());
int readCount = 0;
byte[] data = null;
MemoryStream ms = new MemoryStream();

do
{
  // 將下載資料流的資料讀出, 並寫入 MemoryStream 中。
  data = new byte[8192];
  readCount = reader.Read(data, 0, 8192);
  ms.Write(data, 0, readCount);
}
while (readCount > 0);
```

```
ms.Flush();
ms.Position = 0;
reader.Close();
response.Close();
// 將檔案輸出到瀏覽器。
Response.AddHeader("Content-Disposition", string.Format(
  "attachment; filename=" + this.cboFileList.SelectedValue));
Response.HeaderEncoding = System.Text.Encoding.Default;
Response.BinaryWrite(ms.ToArray());
Response.End();

ms.Close();
ms.Dispose();
```

Q54 我要如何在 ASP.NET 中存取網路磁碟 (或分享資料夾) 中的檔案？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

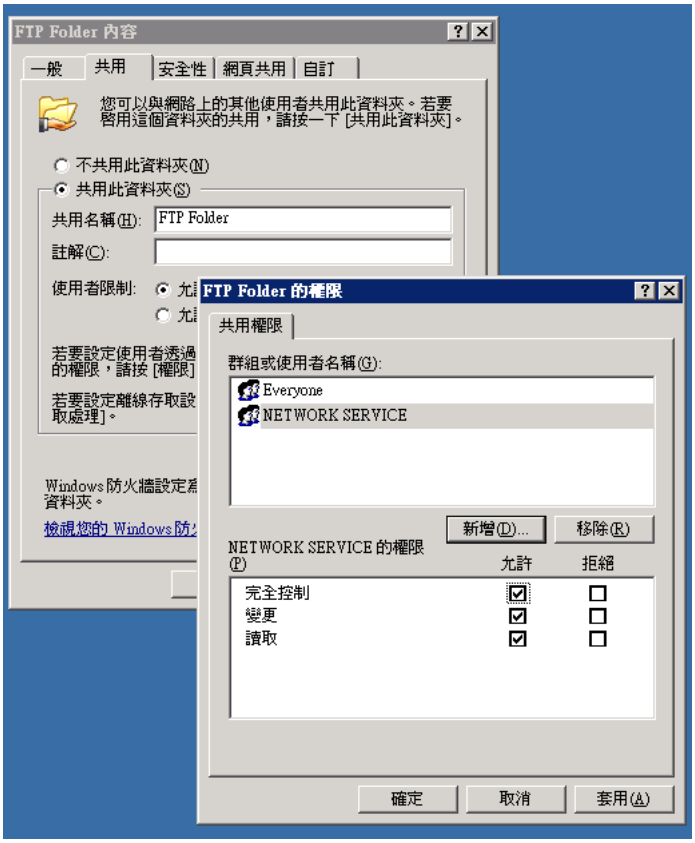
問題

我用 ASP.NET 發展了公司的文件管理系統, 最近公司新購了一台檔案伺服器, 並且要求我要將檔案寫入到新的檔案伺服器中 (有提供分享資料夾) , 請問我要如何做, 才能將上傳的檔案寫入到檔案伺服器的分享資料夾中？

問題說明

筆者在「如何將圖檔儲存在資料庫中」問答中, 建議公司的文件管理 (或知識管理) 系統, 應該將檔案儲存到資料庫中比較安全, 然而若存到資料庫確有其困難 (例如資料庫伺服器老舊, 可能無法負荷大型資料庫的讀寫, 或是磁碟空間預期會不足等) , 可考慮使用檔案伺服器來儲存, 並且放在與一般網路隔離的網路空間中, 並由網管人員監控, 如此可以保護檔案伺服器不受外部網路的干擾, 主控權也可以回到應用程式中, 由應用程式來決定檔案的存取權。

那這就會有個問題了, 依照預設的情況, 分享資料夾是不允許寫入, 除非檔案伺服器開放權限, 且若分享資料夾開放的權限比較高, 可能 ASP.NET 應用程式會無法存取 (ASP.NET 的執行帳戶只有可供執行應用程式的最小權限) , 因此在設定分享資料夾時, 需要確實的允許 ASP.NET 應用程式可以存取分享資料夾, 否則會出現拒絕存取的錯誤訊息。



只要分享資料夾的權限部份沒有問題的話, 那 ASP.NET 讀寫分享資料夾的程式就很簡單了, 只要利用 System.IO 命名空間中的 FileStream 類別即可讀寫分享資料夾中的檔案, 利用 DirectoryInfo 類別可以瀏覽與搜尋分享資料夾中的檔案, 一切的操作方法就和在本機上的檔案相同了。

```
// 讀取網路分享資料夾的檔案, 並輸出到瀏覽器。
FileStream fs = new FileStream(@"\\FileServer\Path\MyFile.docx",
  FileMode.Open, FileAccess.Read);
byte[] data = new byte[(int)fs.Length];
fs.Read(data, 0, (int)fs.Length);
fs.Close();
```

```
Response.BinaryWriter(data);

// 搜尋分享資料夾中的檔案。
DirectoryInfo di = new DirectoryInfo(@"\\FileServer\Path");
FileInfo[] list = di.GetFiles("*.docx");
```

不過如果在檔案讀寫 (尤其是寫入) 時發生網路問題時, 應用程式必須要能夠即時的反應, 將原本寫入資料庫的記錄撤回才行, 否則就會發生資料庫有記錄卻沒有檔案的異常狀況, 所以就必須使用交易 (Transaction) 的功能。

```
using System.IO;
using System.Data;
using System.Data.SqlClient;

...

private void SaveFile(string FileName, byte[] FileContent)
{
try
  {
    // 將檔案儲存到 File Server。
    // //MyServer/FileStore 是範例路徑, 讀者需用自己的路徑取代。
    string fileserverPath = "//MyServer/FileStore";
    FileStream fs = new FileStream(
      fileserverPath + "/" + FileName, FileMode.Create,
      FileAccess.Write);
    fs.Write(FileContent, 0, (int)FileContent.Length);
    fs.Flush();
    fs.Close();
```

```
    // 將檔案記錄儲存到資料庫。
    SqlConnection conn = new SqlConnection(...);
    conn.Open();
    SqlTransaction tran = conn.BeginTransaction();
    SqlCommand cmd = new SqlCommand(
      "INSERT INTO FileList VALUES (@filename, @filepath", conn);
    cmd.Transaction = tran;
    cmd.CommandType = CommandType.Text;
    cmd.ExecuteNonQuery();
  }
  catch (SqlException e)
  {
    // 資料庫寫入時發生錯誤，撤回交易並刪除已儲存的檔案。
    tran.Rollback(); // 撤回交易
    FileInfo fi = new FileInfo(fileserverPath + "/" + FileName);
    fs.Delete(); // 刪除已儲存的檔案。
    fs.Close();
  }
  catch (IOException e2)
  {
    // 網路可能發生問題或是其他 I/O 因素導致無法寫入，回報給使用者。
  }
}
```

解決方案

先設定好檔案伺服器的分享資料夾權限, 並且讓 ASP.NET 的執行帳戶具有可讀寫的權限後, 使用 System.IO 命名空間的類別即可讀寫分享資料夾中的檔案, 以及瀏覽資料夾與檔案等。

Q55 我要如何在 ASP.NET 中抓取來自於其他網站的資料？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

最近公司有一些需求, 要到其他公司提供的網站中抓取數值, 存到公司的資料庫中再取出來分析使用, 網址已經知道了, 那我要如何在 ASP.NET 中去抓取指定網址的資料？

問題說明

相信有些人有這種類似的經驗, 會在桌上放一個自己開發的即時抓取股市數據的小應用程式, 這些數據都來自於一些網站, 像是奇摩股市或是元大證券等網站, 在這些網站上都會提供當日的即時交易數字, 例如每股金額, 漲跌及當日成交量等數字, 這些數字在股市一族的眼中都是可供分析的數字, 在抓取網站的內容後, 再用像 Regular Expression (規則運算式) 來抓取需要的數值存起來, 要查詢時再直接由本地過濾過的資料直接抓取即可。

要抓取網站的內容, 需要經由 Winsock, 利用 HTTP 指令來和 Web Server 溝通, 上傳 HTTP POST 和 HTTP GET, 然後取得伺服器的資料, 上傳的動作稱爲 Request, 而下載的動作稱爲 Response。.NET Framework 中有直接支援 Winsock 網路函式庫的 System.Net.Sockets 命名空間, 有三個主要的類別可用來實作 TCP/IP 網路通訊的功能：

類別	功能
Socket	Winsock 的基底類別, 可用來實作 IPv4, IPv6 的通訊協定。
TcpClient	以 Winsock 實作的 TCP 用戶端。
UdpClient	以 Winsock 實作的 UDP 用戶端。
TcpListener	以 Winsock 實作的 TCP 伺服端。

不過在.NET Framework 還提供了一組透過 HTTP 來存取網站的簡單類別－HttpWebRequest 與 HttpWebResponse, 專門處理使用 HTTP 通訊協定連接到遠端 Web Server, 以抓取資料的類別, 它還有個很類似的類別, 用來存取 FTP 的, 稱爲 FtpWebRequest 與 FtpWebResponse。

解決方案

若想要抓取網頁內容 (例如用奇摩查詢台積電的股票) 時, 可以利用下列程式碼來實作：

程式碼 1：用 HttpWebRequest 開啓簡單的網頁

```
HttpWebRequest request = WebRequest.Create(
  "http://tw.stock.yahoo.com/q/q?s=2330") as HttpWebRequest;
HttpWebResponse response = request.GetResponse() as HttpWebResponse;
StreamReader sr = new StreamReader(response.GetResponseStream(),
  System.Text.Encoding.Default);

strContent = sr.ReadToEnd();

sr.Close();
request = null;
response = null;
```

若遠端網頁是需要登入時, 則需要加上 NetworkCredential, 然後填入帳戶密碼再行連線, 眞正的連線行爲是發生在 HttpWebRequest.GetResponse()時, 所以在這個指令下達前, 所有連線所需要的資訊都要設定好。

若想要知道連線成功或失敗, 可以透過 HttpWebResponse.StatusCode 來取得, 這個屬性會傳回 HTTP 的狀態碼, 常用的幾個狀態碼有：

狀態碼	說明
200	OK
403	拒絕存取
404	找不到網頁
405	設定的 HTTP 動作是不可用的
500	伺服器內部錯誤
503	服務無法使用

若想要使用 HTTP POST 來傳輸資料, 則需要先將傳輸方法設定爲 POST, 然後將要傳輸的資料寫入到 Stream 中 (由 HttpWebRequest.GetRequestStream() 取得) , 然後再下 GetResponse()指令, 例如下列的程式碼：

```
HttpWebRequest request = WebRequest.Create(strURL) as HttpWebRequest;

request.Method = "POST"; // 設定 HTTP POST
request.ContentLength = strPostData;  // 必要項, 設定上傳資料的長度
// 設定爲表單類型。
request.ContentType = "application/x-www-form-urlencoded";

StreamWriter sw = new StreamWriter(request.GetRequestStream());

sw.Write(strPostData); // 寫入要上傳的資料到 Request Stream 中。
sw.Flush();

HttpWebResponse response = request.GetResponse() as HttpWebResponse;
StreamReader sr = new StreamReader(response.GetResponseStream(),
  System.Text.Encoding.Default);

strResponse = sr.ReadToEnd();
```

```
sr.Close();
sw.Close();
request = null;
response = null;
```

下載後的資料解析，就需要花較多的腦筋了，可以試著使用 MSXML (XmlDocument, 因爲 HTML 其實也是一種 XML, 不過若抓取的網站如果不符合 XHTML 規格, 或者本身的 HTML 的用法很亂時, 可能會讓 XmlDocument 擲出例外) 或是 Regular Expression 來做分析, 或是以已知的路徑方式來搜尋資料。

小常識　不要隨便抓取未經授權的網站內容, 小心挨告

HttpWebRequest 與 HttpWebResponse 固然好用, 不過由於用這種方法抓取的內容通常都是未經過合法授權的, 有可能會違反著作權法第 91 條的重製罪：

> 擅自以重製之方法侵害他人之著作財產權者，處三年以下有期徒刑、拘役，或科或併科新臺幣七十五萬元以下罰金。意圖銷售或出租而擅自以重製之方法侵害他人之著作財產權者，處六月以上五年以下有期徒刑，得併科新臺幣二十萬元以上二百萬元以下罰金。以重製於光碟之方法犯前項之罪者，處六月以上五年以下有期徒刑，得併科新臺幣五十萬元以上五百萬元以下罰金。著作僅供個人參考或合理使用者，不構成著作權侵害。

因此, 在抓取別的網站的資料時, 最好是能夠有原作者的書面授權, 以保障自身權益。

Q56 我要如何在 ASP.NET 中送出手機簡訊？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我使用 ASP.NET 來開發網站, 目前我已經完成了一個會員系統, 現在我想要新增使用簡訊查詢密碼的方法, 我要如何由我的程式發出簡訊呢？

問題說明

現在講到手機已經不再稀奇了, 幾乎是人手一支以上, 根據交通部 95 年度爲止的統計數字, 台灣的手機門號戶數爲 23, 249, 262 人, 以內政部 95 年所統計的人口數 22, 876, 527 來計算, 等於每人持有 1.01 支手機, 所以手機已經成爲現代人的主要通訊工具。同時這也給了企業在推廣與行銷產品時的一個廣告的管道, 手機的簡訊功能 (SMS) 除了是一般傳輸簡短訊息聯絡之外, 也成爲了廣告商以及各大行動業者的促銷與廣告的主要管道之一。

一般企業若想要透過應用程式來使用簡訊功能, 都要向電信業者申請, 取得一組簡訊特碼 (SMS Code) , 並且由電信公司提供簡訊服務的管道, 才可以由應用程式產生與發送簡訊, 而簡訊服務的管道視電信業者的不同而有所不同。

電信業者	介面	工具
中華電信	Winsock, TCP	Winsock Library
台灣大哥大	HTTP	HttpWebRequest
遠傳電信	HTTP + MIoD 工具箱	HttpWebRequest

在申請好簡訊特碼後, 電信業者會提供一份說明, 包括 API 的使用, 以及附屬的工具集, 例如遠傳是使用 MIoD 代理伺服器, 台灣大哥大提供 HTTP 通訊的說明文件, 而中華電信則是提供簡訊協定的資料結構與 socket 的 C 範例程式碼供使用。

以.NET 的功能來看, 遠傳電信和台灣大哥大的簡訊發送因爲是使用 HTTP 協定, 故可以利用 HttpWebRequest 與 HttpWebResponse 的組合來執行發送工作, 但中華電信的簡訊發送必須要透過 Winsock 介面, 以 TCP 傳送簡訊的資料結構, 在筆者開發當時確實也吃了不少苦頭, 之後中華電信可能被抱怨太多了 ☺, 在 IMS (整合訊息服務) 中提供了 HTTP 的通訊方法, 也可以利用 HttpWebRequest 來做通訊, 但仍然要按照標準的訊息格式。

在筆者於 2005 年主講程式設計俱樂部的 ASP.NET 實務問答場次中, 曾展示過直接發送簡訊到現場學員的手機中 (當時是可行的) , 現在筆者將這些押箱寶的程式碼公開出來, 供讀者們參考使用, 不過現在可能會因爲協定改變, 或是通訊方法改變, 這些程式碼也許需要修改, 或者參考新的技術規格重新編寫, 筆者由於沒有簡訊特碼可以用, 所以無法在現階段做進一步的測試, 請各位讀者注意。

程式 1：台灣大哥大的簡訊傳送程式 (刪減版)

```
// 簡訊欄位。
private string _userID = null; // required.
private string _password = null; // required.
private string _srcaddr = null; // required.
private string _destaddr = null; // required.
private string _smbody = null; // required.
private string _encoding = null; // optional.
private string _wapurl = null; // optional.
private string _dlvtime = null; // optional.
private string _vldtime = null; // optional.
private string _responseUrl = null; // optional.

public struct Encoding
{
  public const string ASCII = "ASCII";
  public const string BIG5 = "BIG5";
  public const string CP950 = "CP950";
```

```
  public const string NHEAD = "NHEAD";
  public const string PASCII = "PASCII";
  public const string PBIG5 = "PBIG5";
  public const string PUSH = "PUSH";
  public const string UPUSH = "UPUSH";
  public const string UCS2 = "UCS2";
  public const string GB2 = "GB2";
  public const string L_ASCII = "LASCI";
  public const string L_BIG5 = "LBIG5";
  public const string L_NHEAD = "NHEAD";
}

// HTTP URL 範本字串。
private string _templateUrl = "http://{0}:{1}/send.cgi?{2}";
private string _host = null;
private int _port = 0;

public Tcc(string Host, int Port, string UserID, string Password,
  string SrcAddr, string DestAddr, string encoding, string Body)
{
  this._host = Host;
  this._port = Port;
  this._userID = UserID;
  this._password = Password;
  this._srcaddr = SrcAddr;
  this._destaddr = DestAddr;
  this._encoding = encoding;
  this._smbody = Body;
}
```

```
public void SetDeliveryDateTime(DateTime DeliveryDateTime)
{
  this._dlvtime = DeliveryDateTime.ToString("yyyy/MM/dd HH:mm:ss");
}

public void SetValidLimitDateTime(DateTime ValidLimitDateTime)
{
  this._vldtime = ValidLimitDateTime.ToString("yyyy/MM/dd HH:mm:ss");
}

public string SendMessage()
{
  HttpWebRequest request =
    (HttpWebRequest)HttpWebRequest.Create(this.BuildUrl());
  HttpWebResponse response = null;
  Stream stream = null;
  StreamReader reader = null;
  string Result = null;

  request.UserAgent = "TCC_SMS_Manager";
  request.Method = "GET";

  response = (HttpWebResponse)request.GetResponse();

  // 讀取回傳資訊
  stream = response.GetResponseStream();
  reader = new StreamReader(stream);

  Result = reader.ReadToEnd();
  // 釋放資料流
```

```
reader.Close();
response.Close();
response = null;
request = null;

// 檢查回傳值。
// 格式： msgid=value1\nstatusstr=value2
// parse string.
if (Result != null)
{
  string MsgID = Result.Split('\n')[0].Split('=')[1];
  string StatusStr = Result.Split('\n')[1].Split('=')[1];

  if (Int32.Parse(MsgID) > 0)
  {
    // 已傳送, 傳回簡訊的 ID。
    return MsgID;
  }
  else
  {
    switch (MsgID)
    {
      // 請自行依錯誤碼處理錯誤。
    }
  }
}
else
{
```

```
    // 無結果。
    throw new InvalidOperationException(
      "台灣大哥大簡訊主機未傳回任何值 .");
  }
}

private string BuildUrl()
{
  // default RATE_PLAN is A.
  string template =
"username={0}&password={1}&rateplan=A&srcaddr={2}&dstaddr={3}&smbody={4}";
  string strResult =
    string.Format(template, this._userID, this._password,
      this._srcaddr, this._destaddr, this._smbody);

  if (_encoding != null)
    strResult += string.Format("&encoding={0}", this._encoding);
  if (_wapurl != null)
    strResult += string.Format("&wapurl={0}", this._wapurl);
  if (_dlvtime != null)
    strResult += string.Format("&dlvtime={0}", this._dlvtime);
  if (_vldtime != null)
    strResult += string.Format("&vldtime={0}", this._vldtime);
  if (_responseUrl != null)
    strResult += string.Format("&response={0}", this._responseUrl);

  // 回傳套用完成的 HTTP 要求網址
  return string.Format(this._templateUrl, this._host, this._port,
    strResult);
}
```

程式 2：中華電信簡訊傳送程式 (刪減版)

```
// 這是中華電信在只有 Winsock 介面時筆者所撰寫的程式碼
// 現在因爲有 HTTP 介面, 讀者選用 HTTP 的方式來實作
// 目前只測試過傳送英文, 若要傳送中文則要再加以測試修改
// 這裡的 TcpClient 是筆者另外撰寫的 TcpClient 強化工具, 原始碼列於下方

private TcpClient _client = null;
private NetworkStream _ns = null;
private string _userID = null;
private string _password = null;
private string _sendPhoneNumber = null;
private string _primaryServer = null;
private string _backupServer = null;

// 簡訊協定的常數
public struct Constants
{
  public const int MAX_ID_LEN = 8;
  public const int MAX_PASSWD_LEN = 8;
  public const int MAX_MSISDN_LEN = 12;
  public const int MAX_MESSAGEID_LEN = 8;
  public const int MAX_MSG_LEN = 159;
  public const int ORDERTIME_LEN = 12;
  public const byte SERV_CHECK = 0;
  public const byte SERV_EDIT_PASSWD = 1;
  public const byte SERV_SEND = 2;
  public const byte SERV_QUERY = 3;
  public const byte SERV_GET = 4;
  public const byte SERV_SEND_WITH_UDHI = 6;
  public const byte SEND_NOW = 100;
  public const byte SEND_ORDER = 101;
}
```

```
// 簡訊協定－傳訊結構與處理成員
public struct SendMsg
{
  public byte type;
  public byte coding;
  public byte length;
  public byte tran_type;
  public char[] pchID;
  public char[] pchPasswd;
  public char[] pchMsisdn;
  public char[] pchMessageID;
  public byte[] pchMessage;
  public char[] pchSendTime;

  public byte[] ToBytes()
  {
    int length = 10 + Constants.MAX_ID_LEN +
      Constants.MAX_PASSWD_LEN + Constants.MAX_MSISDN_LEN +
      Constants.MAX_MESSAGEID_LEN + Constants.MAX_MSG_LEN +
      Constants.ORDERTIME_LEN;
    byte[] buffer = new byte[length];
    int iBitLocation = 0;

    // fill information.
    buffer[0] = this.type;
    buffer[1] = this.coding;
    buffer[2] = this.length;
    buffer[3] = this.tran_type;

    iBitLocation = 4;
```

```
// pchID
for (int i=0; i<Constants.MAX_ID_LEN+1; i++)
{
  if (i < this.pchID.Length)
    buffer[iBitLocation] = (byte)this.pchID[i];
  else
    buffer[iBitLocation] = 0x0;

  iBitLocation++;
}

// pchPasswd
for (int i=0; i<Constants.MAX_PASSWD_LEN+1; i++)
{
  if (i < this.pchPasswd.Length)
    buffer[iBitLocation] = (byte)this.pchPasswd[i];
  else
    buffer[iBitLocation] = 0x0;
  iBitLocation++;
}

// pchMsisdn
for (int i=0; i<Constants.MAX_MSISDN_LEN+1; i++)
{
  if (i < this.pchMsisdn.Length)
    buffer[iBitLocation] = (byte)this.pchMsisdn[i];
  else
    buffer[iBitLocation] = 0x0;
  iBitLocation++;
}
```

```
    // pchMessageID
    for (int i=0; i<Constants.MAX_MESSAGEID_LEN+1; i++)
    {
      if (i < this.pchMessageID.Length)
        buffer[iBitLocation] = (byte)this.pchMessageID[i];
      else
        buffer[iBitLocation] = 0x0;
      iBitLocation++;
    }

    for (int i=0; i<Constants.MAX_MSG_LEN+1; i++)
    {
      if (i < this.pchMessage.Length)
        buffer[iBitLocation] = (byte)this.pchMessage[i];
      else
        buffer[iBitLocation] = 0x0;
      iBitLocation++;
    }

    // pchSendTime
    for (int i=0; i<Constants.ORDERTIME_LEN+1; i++)
    {
      if (i < this.pchSendTime.Length)
        buffer[iBitLocation] = (byte)this.pchSendTime[i];
      else
        buffer[iBitLocation] = 0x0;
      iBitLocation++;
    }

    return buffer;
  }
}
```

```
// 簡訊協定－接收訊息結構，用來接收簡訊伺服器的回應。
public struct RecvMsg
{
  public byte code;
  public byte coding;
  public byte length;
  public char[] send_msisdn;
  public char[] recv_msisdn;
  public char[] buffer;
}

public CHT(string ServerAddr, int Port, string SendPhoneNumber)
{
  this._client = new TcpClient(ServerAddr, Port); // 初始化 TcpClient
  this._sendPhoneNumber = SendPhoneNumber;

  this._client.SendTimeout = 100;
  this._client.ReceiveTimeout = 100;
}

public CHT(string ServerAddr, string BackupServerAddr, int Port)
{
  this._client = new TcpClient(ServerAddr, Port);

  this._client.SendTimeout = 100;
  this._client.ReceiveTimeout = 100;

  this._primaryServer = ServerAddr;
  this._backupServer = BackupServerAddr;
}
```

```
~CHT()
{
  if (this._ns != null)
    this._ns = null;

  if (this._client != null)
    this._client = null;
}

// helper.
private IPEndPoint CreateEndPoint(string RemoteAddress, int Port)
{
  byte[] ip_addr = new byte[4];
  IPAddress ipaddr = null;
  IPEndPoint ep = null;

  ipaddr = IPAddress.Parse(RemoteAddress);
  ep = new IPEndPoint(ipaddr, Port);

  return ep;
}

// 初始化傳訊結構。
public SendMsg InitSendMsg()
{
  SendMsg msg = new SendMsg();

  msg.pchID = new char[Constants.MAX_ID_LEN+1];
  msg.pchPasswd = new char[Constants.MAX_PASSWD_LEN+1];
  msg.pchMsisdn = new char[Constants.MAX_MSISDN_LEN+1];
```

```
  msg.pchMessageID = new char[Constants.MAX_MESSAGEID_LEN+1];
  msg.pchMessage = new byte[Constants.MAX_MSG_LEN+1];
  msg.pchSendTime = new char[Constants.ORDERTIME_LEN+1];

  return msg;
}

// 初始化接收訊息結構
public RecvMsg InitRecvMsg()
{
  RecvMsg msg = new RecvMsg();

  msg.send_msisdn = new char[Constants.MAX_MSISDN_LEN+1];
  msg.recv_msisdn = new char[Constants.MAX_MSISDN_LEN+1];
  msg.buffer = new char[Constants.MAX_MSG_LEN+1];

  return msg;
}

public void Open()
{
  // connect to remote server.
  // detect socket error, if connection refused,
  // try connection for 2nd IP.
  try
  {
    this._client.Open();
  }
  catch (SocketException se)
  {
```

```
    if (se.ErrorCode == 10061)
    {
      try
      {
        // try 2nd IP for connection.
        this._client.Open(this._backupServer);
      }
      catch (SocketException socketEx)
      {
        throw new Exception(string.Format(
          "Backup IP connection Socket Error Code: {0}",
          socketEx.ErrorCode));
      }
      catch (Exception ex)
      {
        throw new Exception(
          "Backup IP connection error : " + ex.Message + "\n\n" +
          ex.StackTrace);
      }
    }
    else
      throw new Exception(
        string.Format(
          "Primary IP connection Socket Error Code: {0}",
          se.ErrorCode));
  }
  catch (Exception ex)
  {
    throw new Exception("Primary IP connection error: " +
      ex.Message + "\n\n" + ex.StackTrace);
  }
}
```

```
public void Close()
{
  // close remote connection.
  this._client.Close();
}

public void Send(SendMsg sendMsg)
{
  // send command.
  this._client.SendStructure(sendMsg);
}

// 接收簡訊伺服器的回傳訊息。
public RecvMsg Recv()
{
  byte[] data = this._client.ReceiveRawData();
  RecvMsg msg = this.InitRecvMsg();
  string Message = string.Empty;
  int size = 200;
  char[] buf = ASCIIEncoding.ASCII.GetChars(data, 0, data.Length);

  // 將值送入結構中。
  if (buf.Length != 0)
  {
    msg.code = data[0];
    msg.coding = data[1];
    msg.length = byte.Parse(data[2].ToString());
    msg.send_msisdn = this.GetChars(buf, 3, msg.send_msisdn.Length);
    msg.recv_msisdn = this.GetChars(buf, 3 +
      msg.send_msisdn.Length, msg.recv_msisdn.Length);
```

```
    msg.buffer = this.GetChars(buf,
      3 + msg.send_msisdn.Length + msg.send_msisdn.Length,
      size - (3 + msg.send_msisdn.Length+msg.recv_msisdn.Length));

    if (msg.code == (byte)12) // 檢查 msg.code 值。
    {
      // build message string.
      for (int i=0; i<data.Length; i++)
        Message += data[i].ToString();

      throw new InvalidOperationException(
        string.Format("Format Error: {0}", Message));
    }
  }
  else
    throw new FormatException("Data Length: 0");

  return msg;
}

// helper.
// parse byte[] to structure.
public char[] GetChars(char[] Source, int Start, int Length)
{
  char[] result = new char[Length];
  for (int i=Start; i<Length; i++)
    result[i] = Source[i];

  return result;
}
```

```
private void ParseNullBytesToEmptyString(ref byte[] Data)
{
  for (int i=0; i<Data.Length; i++)
  {
    // set byte=0 to String.Empty.
    // code : 0x20,
    if (Data[i] == 0x00)
      Data[i] = 0x20;
  }
}

// public functions.
// use this to do your operation.
public void Login(string UserID, string Password)
{
  SendMsg sendMsg = this.InitSendMsg();
  RecvMsg recvMsg = this.InitRecvMsg();
  string RawData = null;

  this._userID = UserID;
  this._password = Password;

  // 設定登入簡訊伺服器的資訊。
  sendMsg.type = Constants.SERV_CHECK;
  sendMsg.pchID = UserID.ToCharArray();
  sendMsg.pchPasswd = Password.ToCharArray();

  this._client.Send(sendMsg.ToBytes()); // 執行登入。
```

```
  recvMsg = this.Recv(); // 接取登入結果。

  switch (recvMsg.code)
  {
    case 0:
      // authenticated successfully.
      break;

    case 1:
      throw new Exception(RawData);

    case 2:
      throw new Exception("Cannot send from this IP.");

    case 3:
      throw new Exception("System error, try again.");

    case 4:
      throw new Exception("This account is forbidden.");
  }
}

// 傳送訊息
public void SendMessage(string TargetPhoneNumber, byte coding,
  string Message, out string MessageID)
{
  SendMsg sendMsg = this.InitSendMsg();
  RecvMsg recvMsg = this.InitRecvMsg();
  string BufferContent = string.Empty;
  char[] pchID = { (char)0x00 };
```

```
// 準備傳送訊息結構。
sendMsg.type = Constants.SERV_SEND;
sendMsg.coding = coding;
sendMsg.pchMsisdn = TargetPhoneNumber.ToCharArray();
sendMsg.pchMessage = Encoding.Default.GetBytes(Message);
sendMsg.length = (byte)sendMsg.pchMessage.Length;
sendMsg.tran_type = Constants.SEND_NOW;

// 執行傳送
this._client.SendBufferSize = sendMsg.ToBytes().Length;
this._client.Send(sendMsg.ToBytes());

// 取得傳送結果
this._client.recvBufferSize = 200;
recvMsg = this.Recv();

// build buffer.
for (int i=0; i<recvMsg.buffer.Length; i++)
  BufferContent += new string(recvMsg.buffer[i], 1);

switch (recvMsg.code)
{
  case 0:
    MessageID = new string(recvMsg.buffer).Trim();
    break;

  case 1:
    throw new Exception("Cannot send character.");

  case 2:
    throw new Exception("Message sending failure.");
```

```
    case 3:
      throw new Exception("Orderd time beyond 48 hours.");

    case 4:
      throw new Exception("Send binary to pager.");

    case 5:
      throw new Exception("Code transfer fail.");

    case 6:
      throw new Exception("Message sequence number error.");

    case 7:
      throw new Exception("Message out of sequence.");

    default:
      throw new Exception(string.Format(
        "Unknow code: {0}, Content: {1}", recvMsg.code,
        BufferContent));
  }
}

// 查詢簡訊狀態。
public bool Query(string TargetPhoneNumber, string MessgaeID)
{
  SendMsg sendMsg = this.InitSendMsg();
  RecvMsg recvMsg = this.InitRecvMsg();

  // 設定查詢指令。
  sendMsg.type = Constants.SERV_QUERY;
```

```
// 傳送。
this.Send(sendMsg);

// 取回結果。
recvMsg = this.Recv();

switch (recvMsg.code)
{
  case 0:
    // message sent successfully.
    return true;

  case 1:
    // message processing.
    return false;

  case 2:
    throw new Exception("Message sending failure.");

  case 3:
    throw new Exception("Orderd time beyond 48 hours.");

  case 4:
    throw new Exception("Send binary to pager.");

  case 5:
    throw new Exception("Code transfer fail.");

  case 6:
    throw new Exception("Message sequence number error.");
```

```
    case 7:
      throw new Exception("Message out of sequence.");

    default:
      throw new Exception(string.Format(
        "Unknow code: {0}, Content: {1}", recvMsg.code,
        recvMsg.buffer));
  }
}

// 接收訊息。
public RecvMsg Receive()
{
  SendMsg sendMsg = this.InitSendMsg();
  RecvMsg recvMsg = this.InitRecvMsg();
  string BufferContent = string.Empty;

  // 設定爲接收訊息指令。
  sendMsg.type = Constants.SERV_GET;

  // 傳送
  this._client.Send(sendMsg.ToBytes());

  // 取回結果
  recvMsg = this.Recv();

  // build buffer.
  for (int i=0; i<recvMsg.buffer.Length; i++)
    BufferContent += new string(recvMsg.buffer[i], 1);
```

```
switch (recvMsg.code)
{
  case 0:
    // get message.
    return recvMsg;

  case 1:
    // 無資料。
    recvMsg = this.InitRecvMsg();
    recvMsg.code = 1;
    return recvMsg;

  case 2:
    throw new Exception("Message sending failure.");

  case 3:
    throw new Exception("Orderd time beyond 48 hours.");

  case 4:
    throw new Exception("Send binary to pager.");

  case 5:
    throw new Exception("Code transfer fail.");

  case 6:
    throw new Exception("Message sequence number error.");

  case 7:
    throw new Exception("Message out of sequence.");
```

```
    default:
      throw new Exception(string.Format(
        "Unknow code: {0}, Content: {1}", recvMsg.code,
        BufferContent));
  }
}

// TcpClient 程式
public class TcpClient : System.Net.Sockets.TcpClient
{
  private NetworkStream _ns = null;
  private string _serverAddr = null;
  private int _serverPort = 0;
  private int _timeout = 5; // 3 sec of timeout.

  public TcpClient(string ServerAddress, int Port)
  {
    this._serverAddr = ServerAddress;
    this._serverPort = Port;
  }

  ~TcpClient()
  {
    base.Dispose(true);
    GC.Collect();
  }

  public int Timeout
  {
    get { return this._timeout; }
    set { this._timeout = value; }
  }
```

```
public int recvBufferSize
{
  get { return base.ReceiveBufferSize; }
  set { base.ReceiveBufferSize = value; }
}

public int sendBufferSize
{
  get { return base.SendBufferSize; }
  set { base.SendBufferSize = value; }
}

public void Open()
{
  base.Connect(this._serverAddr, this._serverPort);
}

public void Open(string ServerAddr)
{
  base.Connect(ServerAddr, this._serverPort);
}

public void Open(string ServerAddr, int Port)
{
  base.Connect(ServerAddr, Port);
}

new public void Close()
{
  base.Close();
}
```

```
// send string directly.
public void Send(string Data)
{
  byte[] rawData = ASCIIEncoding.ASCII.GetBytes(Data);

      if (!base.Active)
    throw new Exception("Connection is not be created.");

  // get network stream.
  this._ns = base.GetStream();
  // write data.
      this._ns.Write(rawData, 0, rawData.Length);
}

// object convert to string and send it.
public void Send(object Data)
{
  byte[] rawData = ASCIIEncoding.ASCII.GetBytes(Data.ToString());

  if (!base.Active)
    throw new Exception("Connection is not be created.");

  // get network stream.
  this._ns = base.GetStream();
  // write data.
  this._ns.Write(rawData, 0, rawData.Length);
}

public void Send(byte[] rawData)
{
```

```
  if (!base.Active)
    throw new Exception("Connection is not be created.");

  // get network stream.
  this._ns = base.GetStream();
  // write data.
  this._ns.Write(rawData, 0, rawData.Length);
}

// object convert to string and send it.
public void SendStructure(object Data)
{
  if (!base.Active)
    throw new Exception("Connection is not be created.");

  // 儲存體。
  // 建立指標。
  IntPtr ptr = Marshal.AllocCoTaskMem(Marshal.SizeOf(Data));
  byte[] rawData = new byte[Marshal.SizeOf(Data)];

  // 轉換結構到指標 (like memset() API.)
  Marshal.StructureToPtr(Data, ptr, true);
  // 複製結構到位元組儲存體。
  Marshal.Copy(ptr, rawData, 0, Marshal.SizeOf(Data));

  // get network stream.
  this._ns = base.GetStream();
  // write data.
  this._ns.Write(rawData, 0, rawData.Length);
}
```

```
public string Receive()
{
  byte[] buffer = this.ReceiveRawData();
  return this.ParseBytesToString(buffer);
}

public byte[] ReceiveRawData()
{
  DateTime sTimeout = DateTime.Now.AddSeconds(this._timeout);
  MemoryStream ms = new MemoryStream();
  byte[] buffer = new byte[base.ReceiveBufferSize];

  // wait specified time for return.
  while (DateTime.Now < sTimeout)
  {
    // do nothing, for loop and exit when timeout.
  }

  this._ns = base.GetStream();

  // receive data.
  while (this._ns.DataAvailable)
  {
    // read to buffer.
    this._ns.Read(buffer, 0, base.ReceiveBufferSize);
    // write to memory stream.
    ms.Write(buffer, 0, buffer.Length);
  }

  ms.Flush();
  ms.Position = 0;
```

```
    byte[] result = new byte[ms.Length];

    // read from memory stream.
    ms.Read(result, 0, result.Length);
    ms.Close();

    return result;
  }

  public object ReceiveStructure(System.Type StructType)
  {
    // 讀取資料並產生資料指標空間。
    byte[] t_buffer = this.ReceiveRawData();
    IntPtr ptr = Marshal.AllocCoTaskMem(t_buffer.Length);
    object Result = new object();

    // 資料複製到指標。
    Marshal.Copy(t_buffer, 0, ptr, t_buffer.Length);
    // 轉換指標資料到結構。
    Result = Marshal.PtrToStructure(ptr, StructType);

    return Result;
  }

  // helper.
  private string ParseBytesToString(byte[] Data)
  {
    for (int i=0; i<Data.Length; i++)
    {
```

```
            // set byte=0 to String.Empty.
      // code : 0x20,
      if (Data[i] == 0x00)
                Data[i] = 0x20;
    }

    // convert to string.
    return
      ASCIIEncoding.ASCII.GetString(Data, 0, Data.Length).Trim();
  }
}
```

程式3：遠傳電信的簡訊傳送程式

```
private HttpWebRequest _request = null;
private HttpWebResponse _response = null;
private string _MiodServletContext = null;
private string _serverAddr = null;
private int _port = 0;

// 帳戶資訊
private string _username = null;
private string _password = null;

// URL 範本。
private string _pattern =
  "http://{0}/{1}/servlet/com.fet.miod.SendSMS?{2}";

public FETNET(string ServerAddr, int Port, string UserName,
  string Password, string MiodServletContext)
{
```

```
  this._serverAddr = ServerAddr;
  this._port = Port;
  this._username = UserName;
  this._password = Password;
  this._MiodServletContext = MiodServletContext;
}

// NOTE: 訊息內容必須用 Server.UrlEncode() 編碼過。
public void Send(string[] Target, string Message, string Language,
  DateTime DeliverDate, int Dr_Flag)
{
  string QueryString = null;
  string Url = null;

  // generate query string.
  QueryString += string.Format("username={0}", this._username);
  QueryString += string.Format("&password={0}", this._password);
  QueryString += string.Format("&language={0}", Language);
  QueryString += string.Format("&message={0}", Message);

  // additional parameters.
  if (!Dr_Flag.Equals(null))
    QueryString += string.Format("&dr_flag={0}", Dr_Flag);
  if (!DeliverDate.Equals(null))
    QueryString += string.Format("&deliver_date={0}",
      DeliverDate.ToString("yyyyMMddhhmmss"));

  // bind target parameter.
  if (Target != null)
  {
```

```
    for (int i=0; i<Target.Length; i++)
      QueryString += string.Format("&target{0}={1}", i, Target[i]);
  }

  Url = string.Format(
  this._pattern, string.Format("{0}:{1}", this._serverAddr,
    this._port),
  this._MiodServletContext, QueryString);

  this._request = (HttpWebRequest)HttpWebRequest.Create(
    string.Format("{0}:{1}", this._serverAddr, this._port));
  // send message.
  this._response = (HttpWebResponse)this._request.GetResponse();

  if (this._response.StatusCode != HttpStatusCode.OK)
  {
    string strCode = this._response.StatusCode.ToString();
    this._response.Close();
    this._request = null;
    this._response = null;
    throw new InvalidOperationException(
      string.Format("HTTP CODE: {0}", strCode));
  }

  this._response.Close();
  this._request = null;
  this._response = null;
}
```

解決方案

首先要向電信業者申請, 取得簡訊特碼與 API 規格之後, 依照其通訊的方式, 使用 HttpWebRequest 或是 Winsock TCP 撰寫通訊程式, 並且測試是否順暢。

小常識　不可濫發廣告簡訊

雖然簡訊是一個新的行銷管道, 但現在由於廣告簡訊實在過於氾濫, 消費者往往對這些廣告視而不見, 或者選擇拒收 (各家電信業者應都有這樣的服務, 可讓使用者拒收廣告簡訊), 不但會使廣告效果大打折扣, 而且可能會因爲未經使用者的授權而導致官司上身, 不可不慎。

Q57 如何連接與存取 Web Service？

適用範圍：V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

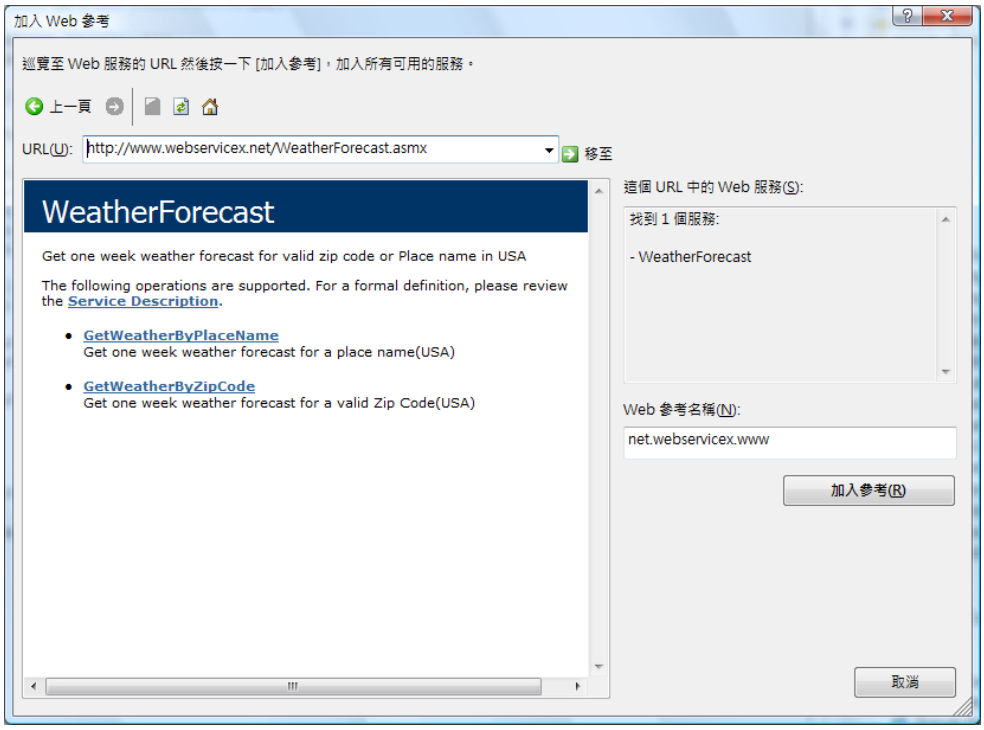
問題

我的公司有一個上游的原料供應商 A 公司, 最近 A 公司和客戶約定要建置 SCM (Supply Chain Management) 系統, 要求向 A 公司採購原物料的廠商, 都要經由他們所開發的 Web Service 訂單管理介面來查詢目前的產能及下單, 請問我要如何連接與存取由他們所提供的 Web Service 介面？

問題說明

簡單的說, Web Service 就是一種掛在網路上, 可連接的函式庫, 並且開放了一些可供呼叫的 API, 用戶端可以透過 SOAP 通訊協定來呼叫, 存取以及使用 Web Service 所開放的服務, 每個服務都由一個方法所組成, 而 Web Service 開放的服務清單與參數, 則由 WSDL 來定義。

在 Microsoft .NET 1.0 時代, 微軟就開始將 Web Service 列入分散式應用程式 (Distributed Application) 的基本要件之一, 與 COM+具有相等的地位。同時, 微軟用 XML Web Service 的名詞來行銷, 其實, XML Web Service 和 Web Service 是指同一種東西。微軟除了在 Microsoft .NET Framework 1.0 中提供了架構 Web Service 所必要的各類支援功能外, 還在 Visual Studio 中加入了一個便利的介面－「加入 Web 參考」, 讓開發人員得以在很簡單的步驟下使用 Web Service。



加入參考的對話方塊會自動解析 URL 的路徑, 將它轉成 Web Service 的參考名稱, 這個名稱即爲 Web Service 在用戶端的代理類別 (Proxy Class) 的命名空間, 而 Web Service 本身的檔案名稱 (例如 WeatherForecast) 就會變成 Web Service Proxy 的類別名稱。

Visual Studio 的加入 Web 參考的功能, 會依照 Web Service 所定義的 WSDL 規格, 將存取 Web Service 所需要的程式碼, 包裝在一個由 Visual Studio 自動產生的程式碼物件中, 這個物件稱爲 Web Service Proxy, 由用戶端的應用程式呼叫, 而 Proxy 物件會自動處理與 Web Service 之間的呼叫以及資料交換, 所以在 Web Service 的使用上, 會有一些資料型別的限制。

只要將 Web Service 加入參考後, 使用的方式就和使用類別物件一樣。

```
private void Button1_Click(object sender, EventArgs e)
{
  // 建立 Web Service Proxy 的物件
  net.webservicex.www.WeatherForecast wf =
    new net.webservicex.www.WeatherForecast();
  // 取得西雅圖 (Seattle) 的一週天氣資料
  string data = wf.GetWeatherByPlaceName("Seattle");
  // 解析並取得資料
  ...
}
```

就用戶端的角度來說, 消費與使用 Web Service 是很簡單的一件事, 如果用 Visual Studio 來實作, 則會更簡單。但有時候會需要自己實作連接的程式 (例如 AJAX) , 則複雜度就會大大的提升, 不過目前網路上已經有很多的 WSDL 與 JavaScript 存取 Web Service 的封裝元件出現, 而且不少是免費的, 因此可以降低用程式碼自實作存取 Web Service 的複雜度, 例如筆者曾使用過的 JavaScript SOAP Client 或是 jQuery 等元件, 下列的程式碼即爲使用 JavaScript SOAP Client 元件連接 Web Service 的 JavaScript 程式碼。

```
using System.Data;
using System.Data.SqlClient;
using System.Xml;

...

[WebMethod]
public string GetProductInfo()
{
  SqlConnection conn = new SqlConnection(
    "Initial Catalog=Northwind; Integrated Security=SSPI");
  SqlCommand cmd = new SqlCommand(
    "SELECT ProductID, ProductName, QuantityPerUnit, UnitPrice FROM
Products", conn);
```

```
  SqlDataReader reader =
    cmd.ExecuteReader(CommandBehavior.CloseConnection);

  XmlDocument doc = new XmlDocument();
  doc.LoadXml("<ProductInfo></ProductInfo>");

  // 將產品的欄位以 XML 的屬性來寫入。
  while (reader.Read())
  {
    XmlNode node = doc.CreateNode(XmlNodeType.Element,
      "Product", null);

    for (int i=0; i<reader.Fields.Count; i++)
    {
      XmlNode attr = doc.CreateNode(XmlNodeType.Attribute,
        "Product", null);
      attr.Name = reader.GetName(i);
      attr.Value = reader.GetValue(i).ToString();
      node.Attributes.SetNamedItem(attr);
    }

    doc.DocumentElement.AppendChild(node);
  }

  reader.Close();
  reader = null;
  cmd.Dispose();
  conn.Dispose();
  return doc.InnerXml;
}
```

```
function GetProductInfo(Barcode)
{
  var pl = new SOAPClientParameters();

  // send request.
  SOAPClient.invoke(
    "http://localhost/MyWebService.asmx",  // Web Service URL
    "GetProductInfo",  // 呼叫的方法
    pl,  // 參數
    true, // 是否要利用非同步通訊
    GetProductWebServiceResponse);  // 非同步的回呼函式
}

function GetProductWebServiceResponse(e)
{
  alert(e); // 顯示由 Web Service 下載的產品清單的 XML 文件檔。
}
```

範例輸出結果 (每個屬性即爲欄位)：

```
<ProductInfo>
  <Product ProductID="1" ProductName="Chai" QuantityPerUnit="10
    boxes x 20 bags" UnitPrice="18.00" />
  <Product ProductID="2" ProductName="Chang"
    QuantityPerUnit="24 - 12 oz bottles" UnitPrice="19.00" />
  ...
</ProductInfo>
```

同樣的, ASP.NET AJAX 也簡化了使用 AJAX 的方式存取 Web Service 的複雜度, 但要記得在 ScriptManager 中註册要使用的 Web Service, 以開放給 ASP.NET AJAX 所用。

```
<asp:ScriptManager ID="myScriptManager" runat="server">
  <Services>
    <asp:ServiceReference  Path="MyWebService.asmx" />
  </Services>
</asp:ScriptManager>
```

解決方案

先取得 A 公司的 Web Service 的 URL 以及 API 文件, 將 Web Service 加入 Web 參考後, 再參考其 API 文件, 運用類別物件的使用方式來呼叫並使用 Web Service 即可。

延伸閱讀

本文所提及的 JavaScript SOAP Library (筆者範例程式中使用的 JavaScript SOAP 用戶端函式庫) 以及 jQuery 用戶端元件, 可以在下列的網址中找到：

- JavaScript SOAP Library：

```
http://www.guru4.net/articoli/javascript-soap-client/en/
```

- jQuery：

```
http://jquery.com/
```

小常識 存取 Web Service 的方法

Web Service 既然是使用 HTTP 的通訊協定, 那麼想當然爾, 用戶端可以利用 HTTP 的 POST 或 GET 方法來連接 Web Service, 但預設的情況下, Web Service 都會認爲用戶端是以 HTTP POST 的方式呼叫。資料交換的部份, Web Service 雖預設是接受 SOAP 格式, 但是仍可以用 HTTP POST 的原始格式呼叫, 所以像是這樣的呼叫也是有效的：

```
HttpWebRequest request = WebRequest.Create(
    "http://www.webservicex.net/WeatherForecast.asmx/
GetWeatherByPlaceName")
  as HttpWebRequest;
request.Method = "POST";
string postData = "PlaceName=Seattle";
ASCIIEncoding encoding = new ASCIIEncoding ();
byte[] byte1 = encoding.GetBytes (postData);
request.ContentType = "application/x-www-form-urlencoded";
request.ContentLength = byte1.Length;
Stream s = request.GetRequestStream ();
s.Write (byte1, 0, byte1.Length);
HttpWebResponse response = request.GetResponse() as HttpWebResponse;
```

不過若想要享有一些進階的功能, 例如 WS-Security 或 WS-Profile 等高階的 Web Service 功能, 除了要引用 Web Service 的強化功能組件 (Web Service Enhancement；WSE) 外, 通訊的資料格式就只能使用 SOAP 來做, 因此 Visual Studio 所扮演的角色就更加重要了。

Q58 如何在 Web Service 中存取 Session？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我正在撰寫公司的一個 Web Service, 用來接收訂單之用, 這個 Web Service 需要驗證, 目前我想要用 Session 來做驗證的工作, 但 Web Service 中可以使用 Session 嗎？又要如何做？

問題說明

Web Service 雖然是一個軟體的元件, 但畢竟它還是得依靠 Web Server 來生存, 因此在 Web Server 上可用的功能, 在 Web Service 中也有部份可以使用。

Session 就是 Web 應用程式中經常會用到的功能之一, 雖然筆者在前面的問答中有強調過不要使用 Session, 但有時候確實會有需要的時候, Web Service 的驗證就是其中之一 (雖然不是最好的方法) , 尤其是要做跨方法運算時, 因爲無法在 Web Service 中使用 ViewState, 所以 Session 就變成了唯一的狀態管理方法了。

若要在 Web Service 的方法中開放使用 Session 的話, 可以在 Web Service 的方法宣告中加上[WebMethod(EnableSession=true)] 的宣告, 例如：

```
[WebMethod(EnableSession=true)]
public string myWebMethod(string arg1, string arg2)
{
  ....
}
```

而 Session 的管理方法, 就和一般的 Web 應用程式完全相同了, 同樣的, 這個方法也可應用在 Application 變數中, 但唯一的差別是, Application 變數不需要經過宣告即可使用。

不過若是要用在使用者驗證的話, 筆者還是建議不要使用比較好, 反而可以利用 SOAP 的 Header 來裝載使用者驗證的資料, 然後在每次的存取中都做驗證, 或者是在資料庫中建立登入的 Token (符記) 資料, 包裝在 SOAP Header 中, 這樣也可以記錄使用者的登入狀況。

SOAP Header 是附加在 SOAP 訊息中的一組資料, 用來讓開發人員可以在 SOAP 訊息中額外加入一些資訊, 讓 Web Service 可以做額外的處理用的, 使用方法很簡單, 首先在 Web Service 程式中, 先宣告一個類別層級的資料類別, 這個類別必須繼承自 SoapHeader 類別：

```
using System.Web.Service;
using System.Web.Service.Protocols;

public class UserInfo : SoapHeader {
  string UserName;
  string Password;
}

public class MyWebService : System.Web.Service.WebService {
    ....
}
```

有了 SOAP Header 的宣告後, 還必須要在 Web Service 的類別中, 加入這個 SOAP Header 的成員變數, 以及屬性存取子, 如此才能夠在 Web Service 中取用這個 SOAP Header 的資料：

```
public class MyWebService : System.Web.Service.WebService {
  private UserInfo _myInfo;
  public UserInfo MyUserInfo
  {
```

```
    get { return this._myInfo; }
    set { this._myInfo = value; }
  }
  ...
}
```

然後, 在要套用這個類別的方法中, 加入[SoapHeader], 的宣告, 包含要存取 SOAP Header 的屬性名稱, 以及 SOAP Header 的資料方向：

```
[WebMethod]
[SoapHeader("MyUserInfo", Direction = SoapHeaderDirection.In)]
public int CalculateSalary(string EmployeeID)
{
  // 取用 SOAP Header 中的資料驗證使用者
  if (!AuthenticateUser(this._myInfo.UserName, this._myInfo.Password))
    return 0;
  ...
}
```

在加入了 SOAP Header 之後, 由於 WSDL 會改變, 因此要重新整理 Web 參考 (若是自行解析 WSDL 的話, 則要修改程式以符合修改過的 WSDL) , 以加入新的 SOAP Header 宣告值。

然後, 在用戶端的部份, 直接新增 SOAP Header 的類別實體, 並且在填入資料後, 設定給 Web Service 所顯露的 SOAP Header 屬性 (即前面所開放的 SOAP Header 屬性) 即可。

```
protected void cmdInvoke_Click(object sender, EventArgs e)
{
  SoapHeaderWebservice ws = new SoapHeaderWebservice();
  // 產生 SOAP Header 的實體。
```

```
  UserInfo userInfo = new UserInfo();
  // 設定 SOAP Header 的值。
  userInfo.UserName = this.T_UserName.Text;
  userInfo.Password = this.T_Password.Text;
  // 設定 SOAP Header 屬性。
  ws.MyUserInfo = userInfo;
  // 呼叫 Web Service。
  Response.Write("<script> alert('" + ws.CalculateSalary(5) +
    "'); </script>");
  ws = null;
}
```

SOAP Header 除了有這樣的功用外, 還有一些其他的用途 (例如輸出 SOAP Header 以回傳額外資料) , 若讀者對這部份有興趣, 可以參考專門講授 Web Service 的書籍, 或者到 MSDN Library 的 XML Web Service 區查詢。

本問題的完整程式如下。

```
// Web Service 程式
using System;
using System.Data;
using System.Data.SqlClient;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;
// 自訂的 SOAP Header
public class UserInfo : SoapHeader
{
  public string UserName;
  public string Password;
}
```

```
public class SoapHeaderWebservice : System.Web.Services.WebService
{
  private UserInfo _myInfo = null; // SOAP Header 變數
  // SOAP Header 屬性。
  public UserInfo MyUserInfo
  {
    get { return this._myInfo; }
    set { this._myInfo = value; }
  }
  // 驗證使用者的程式, 讀者可以自己的驗證程式替換。
  private bool AuthenticateUser(string UserName, string Password)
  {
    if (UserName != "root" && Password != "mypassword")
      return false;
    else
      return true;
  }

  [WebMethod]
  [SoapHeader("MyUserInfo", Direction = SoapHeaderDirection.In)]
  public int CalculateSalary(int EmployeeID)
  {
    if (!AuthenticateUser(this._myInfo.UserName,
      this._myInfo.Password))
      return -1;

    SqlConnection conn = new SqlConnection(
      "Initial Catalog=Northwind; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(@"
      SELECT CAST(ISNULL(SUM(UnitPrice * Quantity * Discount), 0) *
        0.5 as int)
```

```
      FROM Orders o INNER JOIN Employees e ON
        o.EmployeeID = e.EmployeeID
      INNER JOIN [Order Details] od ON o.OrderID = od.OrderID
      WHERE e.EmployeeID = @employeeID", conn);

    cmd.Parameters.Add("@employeeID", EmployeeID);
    cmd.CommandType = CommandType.Text;
    conn.Open();

    int salary = Convert.ToInt32(cmd.ExecuteScalar());
    conn.Close();
    cmd.Dispose();
    conn.Dispose();

    return salary;
  }
}

// 用戶端網頁程式碼，只標示出有使用到 SoapHeader 的程式碼。
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
```

```
public partial class Part5_InvokeMyWebService : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {

  }

  protected void cmdInvoke_Click(object sender, EventArgs e)
  {
    SoapHeaderWebservice ws = new SoapHeaderWebservice();
    // 產生 SOAP Header 的實體。
    UserInfo userInfo = new UserInfo();
    // 設定 SOAP Header 的值。
    userInfo.UserName = this.T_UserName.Text;
    userInfo.Password = this.T_Password.Text;
    // 設定 SOAP Header 屬性。
    ws.MyUserInfo = userInfo;
    // 呼叫 Web Service。
    Response.Write("<script> alert('" + ws.CalculateSalary(5) +
      "'); </script>");
    ws = null;
  }
}
```

解決方案

可修改 WebMethod 的宣告, 設定 EnableSession=true 來啓用 Session 的支援, 在程式中即可使用 Session, 用法和 Web 應用程式的方法相同。

若是不想使用 Session 的話, 可考慮使用 SOAP Header, 一樣可以做到驗證的效果, 唯一要注意的就是 SOAP Header 在網路上傳輸時的安全, 可用 SSL 來將資料流加密, 以保持帳戶的安全性。

考生停看聽

本問題所討論的內容, 可用來準備 Exam 70-529: TS: Microsoft .NET Framework 2.0 Distributed Application Development 的下列主題。

- 組態並客制化 Web Service 應用程式
  - 在 Web Service 中管理工作階段狀態 (Session State)
    - 使用 Session 物件實作工作階段狀態。
  - 實作 SOAP Headers
    - 加入自訂的 SOAP Header 類別。
    - 套用 SoapHeader 屬性於 Web 方法。
    - 在 Web 方法中存取與處理 SOAP Header。
    - 設定 SOAP Header 的輸出入方向。

Q59 如何獲知用戶端的瀏覽器資訊？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我開發了一個線上型錄查詢與瀏覽的程式, 使用者可以在網站上查詢各式各樣的目錄, 不過在某些瀏覽器瀏覽的時候, 會有一些奇怪的問題 (例如：位置位移或是部份特效出不來) , 所以我想要依照瀏覽器的不同而設定一些特效, 或者是讓特效以圖片方式取代, 那麼我要如何取得瀏覽器的設定, 來讓我能夠做這個動作？

問題說明

Web 應用程式本身是一個開放式的平台, 而且也是一組規格的集合, 不但可以讓各家廠商有依據來開發自己的 Web Server, 也可以讓各家廠商團體, 開發自己的瀏覽器, 只要根據由 W3C 所制定的規格 (HTML, XHTML, CSS, Dynamic HTML) 即可, 在行動裝置上亦可以發展瀏覽器與伺服器, 例如 Windows Mobile 的 Pocket Internet Explorer, 以及 Internet 分享器的設定服務等。

目前在市面上主要的瀏覽器佔有率爲：

瀏覽器	佔有率
Internet Explorer	78.68%
Firefox	14.56%
Safari	4.68%
Opera	0.88%
Netscape	0.71%
Opera Mini	0.27%
Mozilla	0.11%

資料來源：Market Share
http://marketshare.hitslink.com/report.aspx?qprid=0

由於每個瀏覽器對於 W3C 的標準與解讀處理的方法都不盡相同, 尤其是在 CSS 版面處理以及 DOM 的規格上, 再加上有些瀏覽器會有其專屬的 HTML 物件 (例如 IE 的 marquee 跑馬燈標籤) , 因此有時候用不同的網頁出來的效果會不同, 網頁設計師和開發人員需要視狀況來處理瀏覽器間差異的問題, 這時就需要知道用戶端所使用的瀏覽器類型, 以及對一些特殊功能的支援。

除了網頁的內容以外, 對於開發人員來說, 瀏覽器的不同有可能會造成功能的無法使用, 最淺顯的一個日子就是 Session, Session 預設的情況是以 Cookie 來做識別, 若瀏覽器沒有支援 cookie 時, 將會讓 Session 失效, 而 Web 應用程式若要使用 cookie 來記錄使用者的設定時, 也會因瀏覽器不支援 cookie 而失效。

需要在伺服器端偵測瀏覽器類型的常見原因有：

- Cookie 的支援。
- 網頁版面的問題。
- 特殊功能支援的問題。
- Scripting 支援的問題。
- ActiveX 或 Plug-in 支援的問題。
- 鎖瀏覽器版本的問題。
- 行動裝置的支援。

要在伺服器端取出用戶端瀏覽器的類型, 以及支援功能的列表, 可以利用 HttpRequest.Browser 屬性, 取得 HttpBrowserCapabilities 物件, 即可以其屬性來取得瀏覽器的類型、版本以及功能支援的資訊。以下的程式碼可以取出較常用到的用戶端瀏覽器資訊：

```
HttpBrowserCapabilities bc = Request.Browser;
Response.Write("<p>Browser Capabilities:</p>");
Response.Write("Type = " + bc.Type + "<br>");
Response.Write("Name = " + bc.Browser + "<br>");
Response.Write("Version = " + bc.Version + "<br>");
Response.Write("Major Version = " + bc.MajorVersion + "<br>");
Response.Write("Minor Version = " + bc.MinorVersion + "<br>");
Response.Write("Platform = " + bc.Platform + "<br>");
Response.Write("Is Beta = " + bc.Beta + "<br>");
Response.Write("Is Crawler = " + bc.Crawler + "<br>");
Response.Write("Is AOL = " + bc.AOL + "<br>");
Response.Write("Is Win16 = " + bc.Win16 + "<br>");
Response.Write("Is Win32 = " + bc.Win32 + "<br>");
Response.Write("Supports Frames = " + bc.Frames + "<br>");
Response.Write("Supports Tables = " + bc.Tables + "<br>");
Response.Write("Supports Cookies = " + bc.Cookies + "<br>");
Response.Write("Supports VB Script = " + bc.VBScript + "<br>");
Response.Write("Supports JavaScript = " + bc.JavaScript + "<br>");
Response.Write("Supports Java Applets = " + bc.JavaApplets + "<br>");
Response.Write("Supports ActiveX Controls = " +
  bc.ActiveXControls + "<br>");
Response.Write("CDF = " + bc.CDF + "<br>");
```

HttpBrowserCapabilities 支援了大多數的瀏覽器, 系統預設支援瀏覽器的定義都存在於.NET Framework 安裝目錄中的 Config\Browsers 目錄中, 以.NET Framework 2.0 來說, 預設支援了 25 種瀏覽器, 同時, .NET 也允許由開發人員自訂 Browser 定義檔, 定義檔是用 XML 的格式, 副檔名是*.browser, ASP.NET 會透過用戶端傳到伺服器的 User-Agent 字串值, 來比對判斷是哪一種瀏覽器, 例如 Internet Explorer 的瀏覽器定義檔中的 User-Agent 字串判斷值爲如下的規則表達式：

```
<browser id="IE" parentID="Mozilla">
  <identification>
    <userAgent match="^Mozilla[^(]*\
      ([C|c]ompatible;\s*MSIE
      (?'version'(?'major'\d+)(?'minor'\.\d+)
      (?'letters'\w*))(?'extra'[^)]*)" />
    <userAgent nonMatch="Opera|Go\.Web|
      Windows CE|EudoraWeb" />
  </identification>
</browser>
```

同時, 若同一種瀏覽器有多個版本, 則在 Browser 定義檔中可以放多個瀏覽器的定義, 並且以額外的 User-Agent 識別字串來判斷：

```
<browser id="IE6to9" parentID="IE5to9">
  <identification>
    <capability name="majorversion" match="[6-9]" />
  </identification>
</browser>
```

小常識　讓 Web 網頁支援行動裝置

由於使用手機上網的人數愈來愈多, 行動裝置瀏覽器 (Mobility Device Browser) 逐漸受到 Web 內容供應商的重視, 由於行動裝置瀏覽器往往無法充份支援一些特殊功能 (例如螢幕色彩, DHTML 或 JavaScript 等), 對於 Web 應用程式來說, 爲了要提供這些瀏覽器適當的內容, 判斷瀏覽器的類型就變得重要了, 另外, ASP.NET 也提供了 Mobile Web Controls 以支援行動裝置上的 HTML 支援 (C-HTML 或 WML 等)。

若想要提供不同的瀏覽器適當的內容, 除了要檢查瀏覽器的類型外, 也可以在 Mobile 控制項中, 針對特定的瀏覽器與裝置建立特定的內容, 這個功能稱爲 Device-Specific Rendering, 它可以允許開發人員用程式化或是單純設定的能力, 設定瀏覽器功能的檢查條件, 然後在 Mobile 控制項中, 套用這個條件以輸出不同的內容。

接下頁

例如, 若想要輸出單色圖片給只支援單色螢幕的瀏覽器, 輸出彩色圖片到支援彩色螢幕的瀏覽器時, 可在 Mobile 的控制項中, 加入對裝置指定的設定：

```
<mobile:Image ID="logoImage" ImageUrl="LogoMono.gif" Runat="server">
  <DeviceSpecific>
    <Choice Filter="isColor" ImageUrl="LogoColor.gif" />
  </DeviceSpecific>
</mobile:Image>
```

<DeviceSpecific>的<Choice>會檢查來自行動裝置瀏覽器的 BrowserCapabilities 物件的指定屬性 (例如 isColor) , 如果值爲 true 時, 就會用<Choice>指定的 ImageUrl 覆寫原先指定的 ImageUrl, 亦即在支援彩色顯示的行動裝置中顯示彩色圖片。

解決方案

利用 Request.Browsers 取得 HttpBrowserCapabilities 物件, 然後透過其屬性取得瀏覽器的類型及各項設定, 若是行動裝置, 則可以再利用 DeviceFilter 及 DeviceSpecific 對特定的裝置支援提供適當的服務。

考生停看聽

本問題所討論的內容, 可用來準備 Exam 70-528: TS: Microsoft .NET Framework 2.0 Web Client Development 的下列主題：

- Web 應用程式的建立與設計
  - Web 應用程式設計
    - 在 Web Form 中偵測瀏覽器類型。
- 建立 ASP.NET Mobile 應用程式
  - 使用 Device-Specific Rendering 來讓控制項顯示在不同類型的行動裝置中。
  - 使用 Mobile Web 控制項將內容在行動裝置上顯示。

Q60 如何存取 Active Directory？

適用範圍：VASP.NET 1.0 VASP.NET 1.1 VASP.NET 2.0 VASP.NET 3.5

問題

目前公司內已經有建置了一個 Active Directory 的網域環境, 我想要在 ASP.NET 程式中處理與列舉出 AD 中的資訊, 請問我要如何利用 ASP.NET 來存取 Active Directory 的資訊？

問題說明

Active Directory 是自 Windows 2000 開始, 就已經存在的網域目錄服務 (Directory Service) , 用來取代 Windows NT 4.0 時代的 NTDS (Windows NT Directory Service) 網域, 它除了具備了 NTDS 應有的能力外, 還延伸支援了 Internet 的相容標準, 例如 Domain Name Service (DNS) 、LDAP (Lightweight Directory Access Protocol) 以及 Kerberos V5 的安全驗證協定等, 讓 Active Directory 可以跨 Internet 提供服務, 而且支援層次化的設計, 以 OU 來切割部門；以 Domain 來劃分公司；以 Forest 來劃分組織的作法, 也讓網域的設計更多了一些彈性。

除了目錄服務基本功能外, Active Directory 亦支援由應用程式開發人員撰寫程式以存取 AD 資料庫的介面, 稱爲 ADSI (Active Directory Service Interface) , 它是一個 COM 元件, 封裝了 AD 的主要介面以及元件等等, 它是與 Active Directory 溝通的介面之一 (另一個是 OLE DB Provider for Active Directory) 。

在 Microsoft .NET Framework 中, ADSI 被獨立封裝在一個命名空間中, 這個命名空間是 System.DirectoryService, 預設情況下是不會加入參考的, 它提供了二個主要類別, 一個是 DirectoryEntry, 另一個則是 DirectorySearcher, 兩者的適用範圍有些許不同, DirectoryEntry 是負責存取物件資料用的, 而 DirectorySearcher 則是負責 AD 的搜尋。

若想要使用 System.DirectoryService 存取 Active Directory, 則要準備好三件事：

- 將 System.DirectoryService.dll 加入參考, 並且在程式中引用 System.DirectoryService 命名空間。
- 準備好可登入網域, 並且具有可以存取到指定資源的權限的使用者帳戶 (包含名稱與密碼) 。
- 將想要查詢的物件階層, 組合成 LDAP 查詢字串。

LDAP 查詢字串是存取 Active Directory 物件的核心, 每一個在 AD 中的物件, 都會有一個固定的唯一名稱, 這個名稱稱爲 Distinguished Names (DN) , DirectoryEntry 會依據 LDAP 所給定的 DN 來搜尋 AD, 若 LDAP 指示的 DN 找不到資料時, 就會傳回伺服器中無此物件的訊息。

LDAP 查詢字串的結構爲：

```
LDAP://[distinguished name]
LDAP://cn=[common name], ou=[Organization Unit Name], dc=[Domain Control Name]
```

在 LDAP 字串中可以使用的名稱代字如表。

字串	代表屬性
DC	domainComponent
CN	commonName
OU	organizationalUnitName
O	organizationName
STREET	streetAddress
L	localityName
ST	stateOrProvinceName
C	countryName
UID	userid

每一個物件都有所屬的結構, 就像是在磁碟機中的檔案一樣, 當要存取檔案時, 都要給完整的路徑字串, AD 也不例外, LDAP 查詢字串就是指定資源位置的字串, 所以也不能有錯誤, 否則會找不到資料。

例如, 若要查詢 Acme.com 的 DC 中的 Administrator 使用者時, 如果只有單一層的 Domain, 那 LDAP 字串就會簡單許多：

```
LDAP://cn=Administrator, cn=Users, dc=acme, dc=com
```

如果要查的是在 Management 的 OU 中的使用者 Steven 的話, 那查詢字串就會變成：

```
LDAP://cn=Steven, ou=Management, dc=acme, dc=com
```

愈複雜的 AD 結構, 這個字串就會愈複雜。

筆者將在「如何查詢 Active Directory 中的使用者或電腦資訊」和「如何用 Active Directory 做使用者驗證」兩個問題中, 深入的討論這二個類別的使用方法。

解決方案

透過System.DirectoryService命名空間的DirectoryEntry與DirectorySearcher類別來取出或搜尋 AD 中的物件, 在使用這兩個類別前, 應先對 AD 的結構以及 LDAP 查詢字串的使用方法有進一步的了解。

Q61 如何查詢 Active Directory 中的使用者或電腦資訊？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

目前公司內已經有建置了一個 Active Directory 的網域環境, 最近公司的管理部門提出了一個需求, 爲了要稽核公司內的使用者狀況, 需要有一個簡單的 Web 存取介面, 讓他們能夠直接看到目前在 Active Directory 中的使用者帳戶與電腦資訊, 請問我要如何利用 ASP.NET 來存取 Active Directory 的資訊？

問題說明

在「如何存取 Active Directory」問答中, 筆者已經大致說明 Microsoft .NET Framework 中的 ADSI 介面類別, 以及 LDAP 查詢字串的各部份元件的意義與組合方法, 接著, 就是要利用它們來瀏覽 Active Directory。

一個 Active Directory 網域, 其組成要件有：

- **網域控制站 (Domain Controller)**：由 domainComponent (dc) 所指定, 負責儲存 AD 資料庫, 與處理來自應用程式的 ADSI 呼叫。
- **網域 (Domain)**：定義了一群 AD 物件組織的邏輯單元, 這個單元可以有很多個, 視網域管理人員的設計而定, 多個網域可以組成一個 Forest。
- **森林 (Forest)**：定義了一個或多個網域, 是組織的最高邏輯單元, 同時可以和其他的森林構建信任關係 (Trust Relationship) , 以相互存取各自 AD 中的資料。
- **組織單元 (Organization Unit)**：由 organizationUnitName (ou) 指定, 代表在 Domain 下的一群 AD 物件, 可能是一個部門, 也可能是一個團隊。
- **AD 物件 (Object)**：代表使用者、電腦或其他的邏輯物件, 一個 AD 物件包含了不定數量的 AD Schema。

- **AD Schema**：這個才是物件的資料儲存單元, 每個物件擁有不同的 Schema, 而且每個物件的 Schema 中儲存的資料也不一定相同。

若想要更進一步的了解 Active Directory 的結構與設計方法, 可參考 Windows Server 2003 的書籍或是 MCSE 的訓練教材, 也可到 TechNet 的網站上查詢。

由於 ADSI 查詢是由上而下, 利用 LDAP 查詢字串來搜尋, 因此 LDAP 字串的下法不可以有錯誤, 否則就會找不到。如果對物件的路徑不是很確定的話, 可以利用 DirectorySearcher 來查詢, 由 DirectorySearcher 可以產生被找到的物件的 DirectoryEntry 物件。

DirectoryEntry 的基本用法爲：

```
DirectoryEntry(LDAP_DN, UserName, Password);
```

LDAP_DN	LDAP 查詢字串, 指定要查詢的物件。
UserName	可開啓由 LDAP 字串所指定物件的使用者帳戶名稱。
Password	可開啓由 LDAP 字串所指定物件的使用者帳戶密碼。

DirectoryEntry 會在每次提取物件資料時, 利用給定的 UserName 和 Password 進行驗證, 如果沒有設定使用者名稱或密碼, 則會用目前登入的使用者帳戶做驗證, DirectoryEntry.AuthenticateType 決定了驗證的方式 (.NET 2.0 開始, 會執行安全驗證, 但若是.NET 1.x, 則預設是不執行驗證) 。

接著, DirectoryEntry 提供了可瀏覽 Schema 的屬性, 以及可變更 Schema 資料的方法：CommitChanges(), 讓開發人員可以在程式中動態的修改 AD 物件的資料, 在 DirectoryEntry 的 Properties 屬性中, 儲存了 AD 物件所屬的 Schema 以及其值, 在取得了 DirectoryEntry 後, 就可利用 DirectoryEntry.InvokeGet() , 或用 DirectoryEntry.Properties[name].Value 來取得 Schema 的值。

```
DirectoryEntry entry = null;
string strLDAP = "LDAP://cn=steven, cn=Users, dc=acme, dc=com";
string strUserName = "myAccount", strPassword = "myPassword";
try
{
  // 開啓 DirectoryEntry,
  // 並且使用 strUserName 與 strPassword 提供的帳戶資訊驗證。
  entry = new DirectoryEntry(strLDAP, strUserName, strPassword);

  // 取出 AD 物件的 Schema
  string[] keys = new string[entry.Properties.Count];
  entry.Properties.PropertyNames.CopyTo(keys, 0);

  for (int i = 0; i < keys.Length; i++)
    Response.Write(
      "Schema: " + keys[i] + " Value: " +
      entry.Properties[keys[i]].Value.ToString() + "<br>");
}
catch (DirectoryServicesCOMException dsex)
{
  Response.Write(dsex.Message);
}

entry = null;
```

若要修改AD物件的Schema的值,則要使用DirectoryEntry.InvokeSet()將值寫入到AD Schema中,但還要記得呼叫DirectoryEntry.CommitChanges(),否則修改不會生效。

```
DirectoryEntry entry = null;

try
{
  entry = new DirectoryEntry(this.T_Url.Text, this.T_UserName.Text,
    this.T_Password.Text);
  // 修改 Schema 的值。
  entry.InvokeSet(this.cboSchema.SelectedValue, this.T_Value.Text);
  // 修改後一定要呼叫，讓修改生效。
  entry.CommitChanges();
}
catch (DirectoryServicesCOMException dsex)
{
  Response.Write(dsex.Message);
}

entry = null;
```

除了可以做修改以外, DirectoryEntry 還支援向上瀏覽 (DirectoryEntry.Parent 屬性)；向下瀏覽(DirectoryEntry.Children 屬性)以及移動物件(DirectoryEntry.MoveTo())和刪除物件與其子物件(DirectoryEntry.DeleteTree())的方法, 開發人員可以善用它來做物件階層的瀏覽。

若想要在物件容器 (例如 OU) 中搜尋特定物件, 就非 DirectorySearcher 物件莫屬了, DirectorySearcher 可以依照物件 Schema 來搜尋符合的 AD 物件, 只要先設定 DirectorySearcher 要啓始搜尋的起點 (DirectorySearcher.SearchRoot 屬性, 需設定 LDAP 查詢字串, 且指向容器物件) , 搜尋類型 (DirectorySearcher.SearchScope 屬性, 可設定只搜尋單層或多層次) 及過濾字串 (DirectorySearcher.Filter, 設定搜尋的 Schema 與其基準) , 就可以利用 DirectorySearcher.FindAll()或DirectorySearcher.FindOne() 來找出符合條件的 AD 物件。

```
DirectorySearcher searcher = null;

try
{
  // 設定要搜尋的起點位置的 DirectoryEntry 物件。
  searcher = new DirectorySearcher(new DirectoryEntry(
    this.T_Url.Text, this.T_UserName.Text, this.T_Password.Text));
  // 執行搜尋。
  SearchResultCollection results = searcher.FindAll();

  if (results.Count == 0)
    Response.Write("No object found");
  else
  {
    foreach (SearchResult result in results)
      Response.Write(result.Path + "<br>");
  }
}
catch (DirectoryServicesCOMException dsex)
{
  Response.Write(dsex.Message);
}

searcher = null;
```

解決方案

首先, 先洽詢 Active Directory 的管理員, 取得目前在 AD 中所儲存的使用者與電腦清單的位置 (如果不清楚, 也可以用 LDAP://RootDSE 來做爲搜尋起始點, 但建議是取得位置, 因爲如果 AD 很大的話, 從 RootDSE 搜尋的時間會比較久), 然後利用 DirectorySearcher 來做搜尋, 例如下列程式。

```
string ldapURL = "LDAP://cn=Computers, dc=acme, dc=com";
DirectorySearcher searcher = new DirectorySearcher(
  new DirectoryEntry(ldapURL, "myAccount", "myPassword"));

// 設定只取出使用者的 AD 物件。
// 若是電腦就是(objectClass=Computer)
searcher.Filter = "(objectClass=User)";

// 搜尋。
SearchResultCollection results = searcher.FindAll();
```

若想要對 ADSI 有進一步的了解, 可到 MSDN 的網站, 搜尋 Active Directory Domain Service 以及 Active Directory Schema 等字串, 即可找到很多參考資料。

小常識　什麼是 Active Directory Schema？

Active Directory 當初的設計就是針對目錄服務的特性來設計的, 而目錄服務的特性就是可以不斷的容納不同的物件, 在網路環境日趨複雜的今日, 這樣的特性相當的重要, 也因爲目錄服務要能夠「廣納百川」, 它的資料結構自然不能夠被寫死, 要能夠隨著加入的物件的需要, 而擴充它的資料結構, 這樣的能力在 AD 中的實作, 就是 AD Schema, 在預設情況下的 AD 的 Schema, 就已經有數百個之譜, 再加上一些微軟的伺服器產品 (例如 Exchange Server) 會對 AD Schema 有所擴充, 可能數量會超過 1000 個, 這些 Schema 裝載著實際的資料數值, 儲存在 AD 的資料庫中, 而且資料值可能會有不同的型態, 有字串；有二進位值；也可能會有物件資料等。

當然, 也許讀者會認爲內建的 AD Schema 不符合需求 (例如, 將薪水資料放在 AD 中) , 不過 AD Schema 是可以擴充的, 只要開發人員有需要, AD 也提供了擴充 Schema 的功能, 在 Active Directory Domain Service 的文件中, 有一章就在說明如何擴充 Schema, 有興趣的讀者可上網查詢參考。

相關問題

- Q60：如何連結 Active Directory？

Q62 如何用 Active Directory 做使用者驗證？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

最近公司開發了一個入口網站, 原本使用資料庫來做驗證, 但最近公司架設了 Active Directory, 公司要求要將內部應用程式的登入, 改用 Active Directory 做驗證, 請問我要如何結合 Active Directory 驗證？

問題說明

筆者在「Q61：如何查詢 Active Directory 中的使用者或電腦資訊」問答中, 曾經說明了 DirectoryEntry 物件在提取 AD 物件的資料時, 都會利用 UserName 和 Password 來做驗證工作, 所以若想要將 Active Directory 驗證和應用程式的 Forms Authentication 整合在一起的話, 就可以利用這個特性來做。

爲了要讓 AD 進行身份的驗證, 開發人員需要利用 DirectoryEntry 查詢在 AD 中的任何屬性資料 (例如使用者名稱或使用者 ID) , 只要有查詢 AD 資料的動作發生時, AD 就會驗證給定的使用者帳戶與密碼, 如果錯誤時, DirectoryEntry 會擲回例外狀況, 開發人員可以抓取這個例外來通知使用者帳戶或密碼錯誤 (可參考 Q61 中對 DirectoryEntry 的操作) 。

首先, 先要由表單中取得使用者輸入的名稱與密碼, 並且將它綁到 DirectoryEntry 物件中, 若想要在驗證時進一步取得使用者的資訊 (例如 uid 或 SID 等資訊) , 則可以和 DirectorySearcher 做整合, 然後啓始搜尋, 若找不到使用者的物件, 則可傳回錯誤, 若找到了資料, 則可以由 SearchResult.GetDirectoryEntry() 來取得使用者的 AD 物件, 並進一步的取得像 objectSid 或者 userPrincipalName 等 AD 物件資料。

```
DirectorySearcher searcher = new DirectorySearcher(
  new DirectoryEntry("LDAP://dc=jcistudio.idv.tw",
    userName, password));
// 設定搜尋符合使用者名稱的使用者。
searcher.Filter =
  "(&(objectClass=User)( sAMAccountName=" + userName + "))";
// 搜尋使用者。
SearchResult result = searcher.FindOne();

if (result == null)
  Response.Write("User is not exist.");
else
{
  // 取得使用者的 SID 資訊。
  object sid =
    result.GetDirectoryEntry().Properties["objectSid"].Value;
}
```

解決方案

可將資料庫的驗證程式碼, 以 DirectoryEntry 與 DirectorySearcher 組合的程式碼替代，並且由它們來執行實際的驗證程序，若想要取得額外資料，則 DirectorySearcher 可以幫助搜尋物件, 並取得資料。

相關問題

- Q60：如何連結 Active Directory？

Q63 如何防止會員資料庫密碼外洩？

適用範圍：V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

公司目前已經有一個會員資料庫, 但現在密碼都是以明碼儲存, 老闆擔心密碼一旦外洩而被盜用, 會影響到公司的形象, 請問我要如何保護這些密碼, 以確保不會外洩？

問題說明

企業的資訊系統之中, 一定少不了會有一些專門用來存放使用者登入資訊的表格, 這個表格中的資料決定了使用者是否可以正常登入, 並且也決定了使用者的權限, 在資料庫的設計上, 如果能將一般性的資訊和機密資訊分開儲存, 會在資料安全上有一些小小的幫助。例如, 將會員和會員帳戶的資料分開儲存, 並且在帳戶資料表中加設一道權限關卡, 如此不但可以減少外洩的風險, 也可以讓想要竊取資料的惡意人士需要花更長的時間, 而公司就會有足夠的應變時間來處理。

不過攻擊者 (Cracker) 的攻擊手法中, 其中有一項, 是利用人的因素來進行的, 攻擊者會利用人際網路以及關係, 伺機竊取公司資訊系統的機密資訊 (例如防火牆密碼、資料庫的管理員密碼或主機的管理員密碼等) , 這種手法稱爲社交工程 (Social Engineering) , 社交工程通常會是最難以防堵的, 除了用公司的政策與罰則來約束具有資料管理權限的人員 (尤其是網管與 DBA) 外, 還可以在資料上做一些手腳, 例如加密 (Encryption) 。

雜湊 (Hashing) 就是一種將資料加密的防護方法, 它是利用雜湊函式 (Hashing Function) 對資料加密處理, 並輸出一組雜湊字串, 而下次要比對時, 只要將輸入字串用相同的方法做雜湊處理, 並且和原先雜湊過的資料做比對, 如果雜湊值相同, 即可視爲和原先的資料相同, 如果有一個字元不對, 那雜湊值的變化就會很大。

雜湊加密法有一個很重要的特性, 就是它是單向 (One-Way) 的加密方法, 因此無法由雜湊值反向計算回原本的資料值, 就算有惡意使用者進入了存放機密資料的資料表, 因爲機密資訊都已經被雜湊處理, 所以也是徒勞無功。

在.NET Framework 中, System.Security.Cryptographics 命名空間存放了數種雜湊加密的方法 (演算法) ：

演算法	加密強度 (位元)
MD5	128
RIPEMD160	160
SHA1	160
SHA256	256
SHA384	384
SHA512	512

加密強度愈強的演算法, 計算出來的雜湊值就會愈長, 所以在設計欄位時, 如果預期要用到較強的演算法, 就要將字串欄位設計的長一些。

使用方法也很簡單, 首先, 先將字串轉換成位元組 (雜湊演算法是用位元組來計算雜湊的) , 然後呼叫雜湊演算法的 ComputeHash() 方法, 取回雜湊後的位元組, 再將位元組轉換回字串即可。

```
using System.Text;
using System.Security;
using System.Security.Cryptography;

...

// 初始化雜湊演算法。
SHA256Managed algorithm = new SHA256Managed();
// 將 ABCDEF 字串轉換爲位元組陣列。
byte[] str = Encoding.ASCII.GetBytes("ABCDEF");
```

```
// 計算雜湊，取回雜湊值的位元組陣列。
byte[] data = algorithm.ComputeHash(str);
// 將雜湊值位元組轉換回字串。
string result = Encoding.ASCII.GetString(data);
algorithm = null;

// 雜湊前：ABCDE
// 雜湊後：?9?????U?/{????NR}??V???{?#?
```

而比對雜湊的方法, 就只要先將輸入字串雜湊, 然後和原本的雜湊字串做字串比對, 如果符合就是正確的, 否則就是失敗的。另外, 雜湊值經過再次雜湊後, 也會產生和原雜湊值不同的字串, 因此雜湊值的字串保存法是安全的方法, 除非在字串比對時沒有做雜湊的步驟。

還有要注意的一點, 雜湊演算法不能隨意更動, 如果先前用了 SHA1 做雜湊, 那就不能改用 SHA256 做雜湊, 不同的雜湊演算法計算出來的雜湊值是不同的, 如果要更換雜湊演算法, 就要將原先的資料都重新雜湊一次, 因此在設計時就要慎選雜湊的演算法, 以免日後困擾。

解決方案

可以修改程式, 將密碼改用雜湊值來儲存, 並且在登入時務必要將輸入字串進行雜湊, 例如程式 1 所示。

程式1：登入時檢查密碼雜湊值的程式

```
using System.Text;
using System.Security;
using System.Security.Cryptography;

...
```

```
// 將密碼的雜湊值由資料庫讀出, 並放到 PasswordHashedString 參數。
public bool IsPasswordMatched(string Password,
  string PasswordHashedString)
{
  SHA384Managed algorithm = new SHA384Managed();

  string hashedStr = Encoding.ASCII.GetString(
    algorithm.ComputeHash(Encoding.ASCII.GetBytes(Password)));

  bool result = (hashedStr == PasswordHashedString);

  algorithm = null;
  return result;
}
```

因爲雜湊值字串本身是亂碼, 所以無法由資料庫中直接查詢密碼, 要另外實作變更密碼的程式, 同樣的, 寫入密碼的程式也需要處理雜湊工作。

考生停看聽

本問題所討論的內容, 可用來準備 Exam 70-536: TS: Microsoft .NET Framework 2.0 Application Development Foundation 的下列主題。

- 利用.NET Framework 的安全性功能強化.NET Framework 應用程式的安全性。
  - 使用 System.Security.Cryptography 命名空間的類別執行加密、解密與雜湊資料 (System.Security.Cryptography 命名空間)
    - 以雜湊爲主的訊息驗證碼 (HMAC)
      - MD5 類別
      - RIPEMD160類別
      - SHA1 類別
      - SHA256 類別
      - SHA384 類別
      - SHA512 類別

Q64 如何將會員區分爲不同群組 (角色)，並針對群組 (角色) 授予權限？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我設計了一個新的會員網站, 允許會員和非會員瀏覽, 但是我想要建立一些只有會員才可以存取的內容, 請問我要如何對使用者做會員等級的驗證？

問題說明

在大多數的商業網站中, 使用者分級的需求與功能都散布在應用程式的各個部份, 因爲不是所有的使用者都應享有相同的功能, 而是會依不同層次的使用者來做切割, 就像管理部門才可以看到的員工薪水表, 在其他部門的使用者就看不到, 或者只有物流員工才能使用的物流介面, 業務部就無法使用等等。

.NET Framework 的安全性基礎服務中, 本身有一個 Principal 的類別, 它可以由開發人員來設定使用者以及所屬的權限, 並且可以在物件建立後, 檢查使用者是否屬於指定的權限, 它被廣泛的用在各個版本的 ASP.NET 與 .NET Framework 應用程式中, 因爲它眞的太好用了。

雖然 ASP.NET 2.0 提供了 Role Service (Role Manager) 供開發人員取用, 但筆者認爲, 若要實作角色管理, 其實不必依賴 Role Service, 自己實作即可, 因爲需要的方法其實不多, 同時也可以相容於未使用 Membership Service 的網站應用程式。

通常開發人員會將資料儲存到資料庫中, 或者是 XML 檔案中, 所以一般的角色管理模組 (Role Management Module) 需要實作幾個方法, 讓資料可以記錄起來, 以利下次的使用：

AddRole(rolename)	加入新角色。
DeleteRole(role_id)	刪除角色。
EditRole(role_id, rolename)	修改角色名稱。
AddMemberToRole(member_id, role_id)	加入角色的成員。
RemoveMember(member_id)	刪除角色成員。
ChangeRole(member_id, new_role_id)	更換成員所屬的角色。
IsInRole(member_id, role_id)	檢查成員是否屬於特定角色。
GetRoles()	取得角色清單。
GetRoleByID(role_id)	取得角色物件。

```
// 一般角色管理模組的參考類別成員。
public abstract class Role
{
  public abstract void AddRole(string RoleName);
  public abstract void DeleteRole(string RoleID);
  public abstract void EditRole(string RoleID, string RoleName);
  public abstract void AddMemberToRole(string MemberID,
    string RoleID);
  public abstract void RemoveMember(string MemberID);
  public abstract void ChangeRole(string MemberID, string RoleID);
  public abstract bool IsInRole(string MemberID, string RoleID);
  public abstract DataTable GetRoles();
  public abstract DataTable GetRoleByID(string RoleID);
}
```

並且在資料庫中建立兩個資料表, 分別放置角色及角色所屬成員的資料, 若使用者已經內含有角色的欄位, 則不必額外建立, 但要把 AddMemberToRole()和 RemoveMember()的程式做適當改寫。

```
-- Role 所需要的資料表
-- 可依自己的情況來改變。
CREATE TABLE Role (
  RoleID uniqueidentifier NOT NULL PRIMARY KEY,
  RoleName nvarchar(50) NOT NULL,
  CreateDate datetime NOT NULL DEFAULT GETDATE()
)
-- 假設使用者的 ID 是 GUID, 可隨需要改變。
CREATE TABLE RoleMembers (
  MemberID uniqueidentifier NOT NULL,
  RoleID uniqueidentifier NOT NULL REFERENCES Role(RoleID)
)
```

以下程式碼爲一般角色模組的參考實作, 供讀者參考修改使用, 因爲程式碼都是簡單的 ADO.NET 配合 SQL 指令的操作, 並未使用到困難或進階的處理方式, 所以筆者就不再加上註解。

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public abstract class Role
{
```

```
  public abstract void AddRole(string RoleName);
  public abstract void DeleteRole(string RoleID);
  public abstract void EditRole(string RoleID, string RoleName);
  public abstract void AddMemberToRole(string MemberID,
    string RoleID);
  public abstract void RemoveMember(string MemberID);
  public abstract void ChangeRole(string MemberID, string RoleID);
  public abstract bool IsInRole(string MemberID, string RoleID);
  public abstract DataTable GetRoles();
  public abstract DataTable GetRoleByID(string RoleID);
}

public class MyRoleManager : Role
{
  public override void AddRole(string RoleName)
  {
    SqlConnection conn = new SqlConnection(
      "initial catalog=TestDB; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(
      "INSERT INTO Role VALUES (NEWID, @roleName, GETDATE())",
      conn);

    cmd.Parameters.Add(
      "@roleName", SqlDbType.NVarChar, 50).Value = RoleName;

    conn.Open();
    cmd.ExecuteNonQuery();
    conn.Close();

    conn.Dispose();
    cmd.Dispose();
  }
```

```
public override void DeleteRole(string RoleID)
{
  SqlConnection conn = new SqlConnection(
    "initial catalog=TestDB; Integrated Security=SSPI");
  SqlCommand cmd = new SqlCommand(
    "DELETE FROM Role WHERE RoleID = @roleID",
    conn);

  cmd.Parameters.Add(
    "@roleID", SqlDbType.UniqueIdentifier, 16).Value =
      new Guid(RoleID);

  conn.Open();
  cmd.ExecuteNonQuery();
  conn.Close();

  conn.Dispose();
  cmd.Dispose();
}

public override void EditRole(string RoleID, string RoleName)
{
  SqlConnection conn = new SqlConnection(
    "initial catalog=TestDB; Integrated Security=SSPI");
  SqlCommand cmd = new SqlCommand(
    "UPDATE Role SET RoleName = @roleName WHERE RoleID = @roleID",
    conn);

  cmd.Parameters.Add(
    "@roleID", SqlDbType.UniqueIdentifier, 16).Value =
      new Guid(RoleID);
```

```
  cmd.Parameters.Add(
    "@roleName", SqlDbType.NVarChar, 50).Value = RoleName;

  conn.Open();
  cmd.ExecuteNonQuery();
  conn.Close();

  conn.Dispose();
  cmd.Dispose();
}

public override void AddMemberToRole(string MemberID,
  string RoleID)
{
  SqlConnection conn = new SqlConnection(
    "initial catalog=TestDB; Integrated Security=SSPI");
  SqlCommand cmd = new SqlCommand(
    "INSERT INTO RoleMembers VALUES (@memberID, @roleID)", conn);

  cmd.Parameters.Add(
    "@memberID", SqlDbType.UniqueIdentifier, 16).Value =
      new Guid(MemberID);
  cmd.Parameters.Add(
    "@roleID", SqlDbType.UniqueIdentifier, 16).Value =
      new Guid(RoleID);

  conn.Open();
  cmd.ExecuteNonQuery();
  conn.Close();
```

```
    conn.Dispose();
    cmd.Dispose();
  }

  public override void RemoveMember(string MemberID)
  {
    SqlConnection conn = new SqlConnection(
      "initial catalog=TestDB; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(
      "DELETE FROM RoleMembers WHERE MemberID = @memberID", conn);

    cmd.Parameters.Add(
      "@memberID", SqlDbType.UniqueIdentifier, 16).Value =
        new Guid(MemberID);

    conn.Open();
    cmd.ExecuteNonQuery();
    conn.Close();

    conn.Dispose();
    cmd.Dispose();
  }

  public override bool IsInRole(string MemberID, string RoleID)
  {
    SqlConnection conn = new SqlConnection(
      "initial catalog=TestDB; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(
      "SELECT MemberID FROM RoleMembers WHERE RoleID = @roleID",
      conn);
    bool isInRole = false;
```

```
    cmd.Parameters.Add(
      "@roleID", SqlDbType.UniqueIdentifier, 16).Value =
        new Guid(RoleID);

    conn.Open();

    SqlDataReader reader =
      cmd.ExecuteReader(CommandBehavior.CloseConnection);

    while (reader.Read())
    {
      if (reader.GetValue(0).ToString().ToUpper() ==
        MemberID.ToUpper())
      {
        isInRole = true;
        break;
      }
    }

    reader = null;
    conn.Dispose();
    cmd.Dispose();

    return isInRole;
  }

  public override void ChangeRole(string MemberID, string RoleID)
  {
    SqlConnection conn = new SqlConnection(
      "initial catalog=TestDB; Integrated Security=SSPI");
```

```
    SqlCommand cmd = new SqlCommand(
      "UPDATE Role SET RoleID = @roleID WHERE MemberID = @memberID",
      conn);

    cmd.Parameters.Add(
      "@memberID", SqlDbType.UniqueIdentifier, 16).Value =
        new Guid(MemberID);
    cmd.Parameters.Add(
      "@roleID", SqlDbType.UniqueIdentifier, 16).Value =
        new Guid(RoleID);

    conn.Open();
    cmd.ExecuteNonQuery();
    conn.Close();

    conn.Dispose();
    cmd.Dispose();
  }

  public override DataTable GetRoleByID(string RoleID)
  {
    SqlConnection conn = new SqlConnection(
      "initial catalog=TestDB; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand(
      "SELECT * FROM Role WHERE RoleID = @roleID", conn);
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataTable table = null;

    cmd.Parameters.Add(
      "@roleID", SqlDbType.UniqueIdentifier, 16).Value =
        new Guid(RoleID);
```

```
    adapter.Fill(table);

    adapter = null;
    conn.Dispose();
    cmd.Dispose();

    return table;
  }

  public override DataTable GetRoles()
  {
    SqlConnection conn = new SqlConnection(
      "initial catalog=TestDB; Integrated Security=SSPI");
    SqlCommand cmd = new SqlCommand("SELECT * FROM Role", conn);
    SqlDataAdapter adapter = new SqlDataAdapter(cmd);
    DataTable table = null;

    adapter.Fill(table);

    adapter = null;
    conn.Dispose();
    cmd.Dispose();

    return table;
  }
}
```

有了角色資料後, 就可以將角色加到 Principal 物件中, 然後在程式中使用了, 套用的方法可以參考「Q65：如何使用程式化的表單驗證」問答。

驗證的部份做好了, 接下來就可以設定各區段的授權設定, 在 Web.config 中, 可以依照想要授權的區域, 做角色的授與或拒絕的授權, 這個授權方法被稱爲 URL 授權 (URL Authorization) , 開發人員可以設定<location>的方式, 來授與特定 URL 所指定的資源的存取權限。

例如, 網站中有一個會員專區, 網址是/members, 那麼就可以在 Web.config 中設定這個網址只能讓會員進入：

```
<configuration>
  <location path="~/members">
    <system.web>
      <authorization>
        <deny users="?"/>
        <allow roles="members" />
        <allow users="Administrator" />
      </authorization>
    </system.web>
  </location>
</configuration>
```

其中：

<deny users="?" />	表示拒絕匿名使用者。
<allow roles="members" />	表示允許角色爲 members 的使用者進入。
<allow users="Administrator" />	表示允許使用者名稱爲 Administrator 的使用者進入。

使用者或角色若有多個時, 可以用逗號分隔的方式來指定。

一個網站可以有多個<location>的設定, 而一個<location>中可以設定多個<allow>或<deny>的清單, 當使用者帳戶被發現是拒絕存取清單中的帳戶或角色時, 就會被導向到登入的網頁 (由<forms>的 loginUrl 指定)。

如果不想要在 Web.config 中設定的話, 在程式中也是可以檢查角色的, 只要使用下列的程式：

```
if (Request.IsAuthenticated)
{
  if (!HttpContext.Current.User.IsInRole("member"))
    Response.Redirect("login.aspx");
}
else
  Response.Redirect("login.aspx");
```

就可以驗證使用者是否爲 member 角色成員之一, 若否, 則強制的轉移到 login.aspx 登入網頁, 以上的程式的效力和前面的<location>設定的效力相同。

解決方案

可將會員帳戶設定爲 member 角色成員, 而一般使用者則不授與角色, 並且在 Web.config 中設定會員區的可瀏覽角色, 並且拒絕匿名的使用者, 或者在程式中檢查角色, 如此就可以確保非會員看不到會員區的資料。

考生停看聽

本問題所討論的內容, 可用來準備 Exam 70-528: TS: Microsoft .NET Framework 2.0 Web-Based Client Development 的下列主題。

- 實作驗證與授權。
  - 使用授權功能以建立已驗證使用者的權利。
    - 使用 URL 授權來限制應用程式的使用者權限分配。

Q65 如何使用程式化的表單驗證 (Code-Based Forms Authentication)？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

ASP.NET 的表單驗證很好用, 但是都是要用單一個網頁來做登入表單, 有很多網站都是將登入做在一個區塊中, 但是用預設的表單登入指令沒辦法做到, 那我應該要如何做呢？

問題說明

ASP.NET 的表單驗證 (Forms Authentication) 是使用 Cookies 來做使用者識別 (User Identity) 的能力, 在預設的情況下, 開發人員可以實作一個登入網頁 (例如login.aspx),然後在網頁中呼叫FormsAuthentication.RedirectFromLoginPage()來讓驗證資訊生效, RedirectFromLoginPage()將驗證資訊的 Cookies 輸出和實作細節都簡化成一個方法, 讓開發人員得以用最簡單的方法得到登入的功能, 但是 RedirectFromLoginPage()簡化的太多, 若要自訂表單登入流程的話, 勢必沒有辦法處理，此時，開發人員需使用表單驗證的其他公用程式，來達到RedirectFromLoginPage()的效果。

ASP.NET 表單驗證的 RedirectFromLoginPage()內部實作了三件事, 讓表單驗證得以生效, 這三件事爲：

- 產生驗證結果的 Cookie。
- 產生 FormsIdentity。
- 建立 GenericPrincipal, 並設定給 HttpContext.Current.User。

首先, FormsAuthentication 類別爲了要持續保存使用者的驗證訊息, 會產生一組驗證的 Cookie, 這組 Cookie 會和現有的 Cookies 一起寫入到用戶端, 只要透過呼叫 SetAuthCookie() 方法即可。

```
FormsAuthentication.SetAuthCookie(user_name, bPersistent,
  CookiePath);
```

參數	型別	說明
user_name	String	登入的名稱，可以是使用者名稱或是其他的文字。
bPersistent	Boolean	是否要建立持久性的 Cookie。
CookiePah	String	Cookie 所使用的 Cookie Path。

有了這個 Cookie 才能夠確定使用者已經經過驗證。

接著, FormsAuthentication 會產生一組識別物件, 稱爲 FormsIdentity, 這個物件會作爲在應用程式中的使用者識別, 有了這個識別後, 在程式中使用 Request.IsAuthenticated 時, 會傳回 true。FormsIdentity 需要使用驗證程式所產生的授權票證 (ticket) , 這組票證包含了一些使用者的資訊, 以及授權票證的有效期限, 與使用者自訂資料等等, 開發人員可以利用建構函數來產生授權票證：

```
FormsAuthenticationTicket ticket = new FormsAuthenticationTicket(
  version, name, issueDate, expirationDate, isPersistent, userData);
```

參數	型別	說明
version	Integer	票證的版本號碼，可以用做版本識別。
name	String	登入名稱，可以是使用者名稱。
issueDate	DateTime	票證核發日期, 可使用 DateTime.Now 來設定。
expirationDate	DateTime	票證到期日期，必須要設定。
isPersistent	Boolean	是否要建立持久性的 Cookie。
userData	String	使用者自訂資料，此欄不可爲空白。

在產生了授權票證後, 就可以使用它來產生 FormsIdentity：

```
FormsIdentity identity = new FormsIdentity(ticket);
```

爲了要保存這個票證, ASP.NET 表單驗證使用了 Cookie 來保存, 若要存取 Cookies, 就需要使用 HttpCookie 類別來產生 Cookie 資料, 並使用 Response.Cookies.Add() 寫入到用戶端中, 做爲使用者識別用, 例如：

```
HttpCookie cookie = new HttpCookie();
cookie.Name = "Cookie's Name";
cookie.Value = "Cookie's Value";
cookie.Expires = DateTime.Now.AddHours(8);
Response.Cookies.Add(cookie);
```

不過由於 HttpCookie 只接受字串, 以及安全性的考量, ASP.NET 提供了 FormsAuthentication.Encrypt()來將票證加密成爲亂碼字串, 如此即可使用 HttpCookie 來寫入 Cookie。

```
HttpCookie cookie = new HttpCookie(
  FormsAuthentication.FormCookName,
  FormsAuthentication.Encrypt(ticket));
Response.Cookies.Add(cookie);
```

最後, 將識別資訊交給 HttpContext.Current.User, 即可大功告成。HttpContext.Current.User 是 ASP.NET 應用程式中用來識別使用者是否驗證成功的基準, 在匿名的情況下, HttpContext.Current.User 是 NULL (VB 是 Nothing) 值, Request.IsAuthenticated 一定只會回傳 false, 只有在有值時, 才會回傳 true。

HttpContext.Current.User 接受實作 IPrincipal 的物件, 例如 WindowsPrincipal (使用 Windows 整合式驗證) 、PassportPrincipal (使用 Passport 驗證) 與 GenericPrincipal (由程式自訂的驗證) 等等, 以表單驗證方法來說, 需要使用程式自訂的驗證方式, 所以需要用到 GenericPrincipal 物件。

GenericPrincipal 物件需要二個參數, 一個是實作了 IIdentity 介面的物件, 另一個則是授權的角色清單：

```
GenericPrincipal gp = new GenericPrincipal(identity, role_list);
```

參數	型別	說明
identity	IIdentity	實作 IIdentity 介面的物件, 在表單驗證可以套用 FormsIdentity 物件。
role_list	String 陣列	授權的角色清單, 爲一字串陣列。

角色清單 (Role List) 是應用程式可授權的角色的列表, 應用程式可以使用 GenericPrincipal.IsInRole() 來判斷使用者是否在授權的角色清單中, 如此可以實作出角色爲主 (Role-Based) 的授權機制。

所以, 當 FormsIdentity 產生之後, 就可以用來產生 GenericPrincipal 物件, 可用下列的程式碼來實作：

```
GenericPrincipal gp = new GenericPrincipal(identity, "".Split(', ');
HttpContext.Current.User = gp;
```

完成了前面的三個工作之後, 再指示瀏覽器執行重導向, 表單驗證就可以生效了：

```
Response.Redirect(Request.RawUrl);
```

解決方案

經過以上的說明後, 程式化表單驗證這個功能的完整程式碼範本, 列表如下：

```
private void cmdLogin_Click(object sender, EventArgs e)
{
```

```
  // 登入驗證的程式碼。
  // 請修改爲自己的驗證程式。
  string userID = Security.Login(
  this.txtUserName.Text, this.txtPassword.Text);

  // 建立驗證 Cookie。
  FormsAuthentication.SetAuthCookie(this.txtUserName.Text,
    FormsAuthentication.FormCookiePath);

  // 建立驗證用票證。
  FormsAuthenticationTicket ticket = new FormsAuthenticationTicket(
    1, this.txtUserName.Text, DateTime.Now,
    Datetime.Now.AddHours(8), false, userID);

  // 建立 FormsIdentity 物件, 準備套用到 Principal。
  FormsIdentity identity = new FormsIdentity(ticket);

  // 將票證保存在 Cookie 中。
  HttpCookie cookie = new HttpCookie(
    FormsAuthentication.FormCookieName,
    FormsAnthentication.Encrypt(ticket));
  Response.Cookies.Add(cookie);

  // 建立 Principal, 並將 FormsIdentity 物件套用進去。
  GenericPrincipal gp =
    new GenericPrincipal(identity, "".Split(', '));
  HttpContext.Current.User = gp;

  // 對瀏覽器做重導向, 讓驗證生效。
  Response.Redirect(Request.RawUrl);
}
```

可以在 ASP.NET 使用者控制項, 或是認爲內建機制不足, 想要擴充能力時使用。

經驗談

程式化的表單驗證可以用來做更多進階的事情, 例如實作自訂的 Active Directory 驗證 (不使用 WindowsIdentity) , 在 ticket 中放更多的資料 (例如權限值或是更多的識別資訊) , 以及實作多於一個的登入角色驗證機制 (即多個登入畫面) 等等, 實務上有許多像是多重登入的能力, 或者是多層次化登入 (第一層爲使用者, 第二層爲管理員等) 能力。

雖然在 ASP.NET 2.0 中有新增了 Membership 能力, 不過在多數案例中, 程式化的表單驗證仍然被許多應用程式與開發人員所採用, 所以開發人員仍需要對此種技術有所了解。

考生停看聽

本主題所討論的內容, 可用以準備 Exam 70-528: TS: Microsoft .NET Framework 2.0 Web Application 考試的主題：

- 使用表單驗證來產生使用者識別資訊。
- 使用授權機制來建立已驗證使用者的權利。

相關問題

- Q62：如何用 Active Directory 做使用者驗證？
- Q63：如何防止會員資料庫密碼外洩？

Q66 如何使用 ASP.NET 寄信？

適用範圍： V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我的 Web 應用程式是一個 CRM 的系統, 給客服和行銷人員使用的, 最近使用者想要加一個功能, 就是透過系統直接發送電子報或是促銷訊息給指定的客戶群, 客戶群名單不一定, 搜尋客戶名單的功能已經有了, 但不知道要如何寄送 Email, 請問有何方法？

問題說明

在系統上直接寄信是很常見的功能, 舉凡電子報、自動回覆、監控通知、定時報表、…這些都需要由系統上直接寄信, 只要公司中有 SMTP Server (例如 IIS 上的 SMTP Virtual Server 或是 Exchange SMTP Virtual Server) , 或者是有外部申請的信箱, 即可透過程式來連接 SMTP Server 發送郵件。

以往在 ASP 時代, 通常會使用到 CDO for NT Server 這個元件, 若想要定時發送報表, 那可能就必須尋求其他的解決方案 (例如自動定時發送郵件的軟體) 。在.NET 時代, .NET Framework 有內建支援 SMTP 發送信件的類別可以使用, 不過.NET 1.x 和.NET 2.0 使用的類別不同。

版本	命名空間
.NET 1.x	System.Web.Mail
.NET 2.0	System.Net.Mail

System.Web.Mail 是以 CDO for NT Server 元件爲基礎發展的類別, 因而只能使用在 Web 上面, System.Net.Mail 是重新以 SMTP 通訊協定來設計的類別, 不再依賴 CDO for NT Server 元件, 因此可以在 Windows Forms 上使用, 不再只能用在 Web 平台上。

要寄信其實很簡單, 只要透過 SmtpClient 設定好要連接的 SMTP Server (有些會需要帳號密碼, 以及設定 SSL 的保護連線) , 然後把信件放到 MailMessage 物件中, 交由 SmtpClient 寄出即可。

常用屬性	說明
EnableSsl	啓用 SSL 連接 SMTP Server 。
Port	SMTP Server 的通訊埠。
Credentials	連接 SMTP Server 需要的驗證物件, 內含帳戶與密碼。
Host	SMTP Server 的 DNS 名稱或是 IP 位址。
Timeout	SMTP Server 連線的逾時長度。

MailMessage 是一個包裝信件訊息內容的物件, 它可以設定收件人 (包含正本、副本及密件副本) 、信件內容 (純文字與 HTML) , 並且可以設定一些信件屬性 (例如郵件的重要性, 以及郵件的標頭設定等等) , 以及加入郵件的附件檔案等。

以下爲一段使用 Gmail 寄信的 SmtpClient 以及 MailMessage 程式碼：

程式碼 1：使用 SmtpClient 經由 Gmail 寄信的程式碼

```
SmtpClient client = new SmtpClient();


client.Host = "smtp.gmail.com";
client.Port = 587;
client.EnableSsl = true;
client.Credentials = new NetworkCredential("my_gmail_account ",
  "my_gmail_password");
client.Timeout = 150;


Dictionary<string, byte[]> attachments =
  new Dictionary<string, byte[]>();
```

```
if (this.T_File.HasFile)
{
  attachments.Add(this.T_File.FileName, this.T_File.FileBytes);
}

MailMessage msg = ProcessMailMessage("sender_mail",
  this.T_SendToList.Text.Split(', '), this.T_CCList.Text.Split(', '),
  this.T_BCCList.Text.Split(', '), false, MailPriority.Low,
  "TestMessage", this.T_Body.Text, attachments);

client.Send(msg);
client = null;

public MailMessage ProcessMailMessage(
  string Sender, string[] To, string[] CC, string[] Bcc,
  bool IsBodyHTML, MailPriority Priority, string Subject,
  string Body, Dictionary<string, byte[]> Attachments)
{
  MailMessage msg = new MailMessage();

  msg.From = new MailAddress(Sender);
  msg.Priority = Priority;
  msg.Subject = Subject;
  msg.Body = Body;
  msg.IsBodyHtml = IsBodyHTML;

  if (!string.IsNullOrEmpty(To[0]))
    if (To.Length > 0)
      for (int i = 0; i < To.Length; i++)
        msg.To.Add(new MailAddress(To[i]));
```

```
  if (!string.IsNullOrEmpty(CC[0]))
    if (CC.Length > 0)
      for (int i = 0; i < CC.Length; i++)
        msg.CC.Add(new MailAddress(CC[i]));

  if (!string.IsNullOrEmpty(Bcc[0]))
    if (Bcc.Length > 0)
      for (int i = 0; i < Bcc.Length; i++)
        msg.Bcc.Add(new MailAddress(Bcc[i]));

  // process attachments
  if (Attachments != null && Attachments.Count > 0)
  {
    string[] keys = new string[Attachments.Count];
    Attachments.Keys.CopyTo(keys, 0);

    for (int i = 0; i < keys.Length; i++)
    {
      MemoryStream ms = new MemoryStream(Attachments[keys[i]]);
      msg.Attachments.Add(new Attachment(ms, keys[i]));
    }
  }

  return msg;
}
```

解決方案

先取出目的客戶的 Email Address, 並且將信件內容用 MailMessage 包裝好後, 再交由 SmtpClient 去寄出信件即可。

小常識 爲何郵件寄不出去？

在使用 System.Net.Mail 的 SmtpClient 寄信的時候, 在某些情況下會有信件寄不出去的狀況發生, 或者是信件被存到寄信佇列 (Send Queue) 中, 由於在現今廣告信 (Spam) 充斥的狀況, 以及太多廣告信利用不知情的單位的 Mail Server 做信件轉傳 (Mail Relay) 的問題, 讓大多數的 SMTP Server 都有設定轉信的門檻, 例如需要驗證或是設定在特定的網域內才能轉傳, 否則會被退回無法寄送。

另一個常見的原因, 若是用 IIS 的 SMTP Server, 但沒有設定 SMTP Virtual Server 的 IP 位址的話, 就會因爲寄不出去而存到寄信佇列中。多數的系統都會透過專屬的 SMTP Server 來寄信, 而且都是有專屬的管理員在管理, 因此, SmtpClient 只能夠寄出郵件, 但若郵件是因爲 SMTP Server 有問題轉寄不出去時, SmtpClient 是無法得知的, 要由 SMTP Server 來檢查問題的所在。

小常識 如何定時發送信件？

定時發送郵件是個有趣也很常出現的應用, 然而 Web 應用程式是一個被動型的平台, 無法自行偵測與執行動作, 也就是 ASP.NET 無法寫出在伺服端定時執行的應用程式, 必須要依賴 Windows 應用程式, 而最常被用到的類型是 Service Application (服務應用程式) , 也只有這類型的程式具有在背景執行的能力, 和一般的 Windows 應用程式不相同。所以若想要寫一個定時發送信件的應用程式時, 就需要用到 Windows Service。

寫這個 Service 也不難, 只要先建立好 Windows Service 應用程式, 然後撰寫 OnStart 的事件常式, 將計時器啓動程式放在這裡, 然後在計時器事件常式中放置檢查與發信程式即可, 最後在 OnStop 事件常式中放置停止計時的程式, 最後把這個服務安裝到 Windows 中, 就可以使用了。

例如這樣的程式：

```
protected override void OnStart(string[] args)
{
  this._Timer = new Timer();
  this._Timer.Elapsed += new ElapsedEventHandler(Timer_Elapsed);
```

接下頁

```
    this._Timer.Interval = 6000000;
    this._Timer.Start();
}
private void Timer_Elapsed(object sender, ElapsedEventArgs e)
{
    // process information.

    // create mail.
    MailMessage msg = ProcessMailMessage(sender, SendToList, CCList,
      BccList, IsBodyHTML, MailPrioritySetup, Subject, Body, null);
    // send mail
    this.SendMail(msg);
}

protected override void OnStop()
{
    this._Timer.Stop();
    this._Timer.Dispose();
}
```

筆者的小建議是, 先把 SMTP 寄信程式都測試好了, 再把它移到 Windows Service 中, 這樣就能省下不少的測試工夫。

考生停看聽

本問題所討論的內容, 可用來準備 Exam 70-536: TS: Microsoft .NET Framework 2.0 Application Development Foundation 的下列主題。

- 利用.NET Framework 來實作服務行程、執行緒及應用程式域。
  - 實作, 安裝與控制服務 (System.ServiceProcess 命名空間)
    - 繼承自 ServiceBase 類別。
- 於.NET Framework 應用程式中實作郵件功能。
  - 自.NET Framework 透過 SMTP Server 寄信電子郵件。
    - MailMessage 類別。
    - MailAddress 與 MailAddressCollection 類別。
    - SmtpClient 類別。
    - Attachment 類別。

Q67 如何在 ASP.NET 中執行需要管理員權限的程式？

適用範圍：V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我使用 ASP.NET 來開發網站, 想要直接在 ASP.NET 程式中存取一些資源 (檔案、程式或資料夾) , 但這些資源都只能透過管理員權限去存取, 我有辦法將 ASP.NET 的執行帳戶改變成管理員帳戶, 然後存取後再還原回原本的帳戶嗎？

問題說明

ASP.NET 的預設執行帳戶是 Network Service (IIS 5.0 則是 ASPNET) , 這個帳戶的權限很低, 只擁有足夠執行 ASP.NET 應用程式的權限而已, 這也是爲了要保護伺服器, 不受惡意的 ASP.NET 程式的影響所致, 在多數的情況下, 不需要給予 ASP.NET 程式執行的帳戶太大的權限, 那只會有一些副作用而已。

不過在一些特殊的案例中, 確實是會需要使用到管理員權限 (或需要提高到足夠的權限) 的, 例如要存取系統保護的資料, 或者是一些被限制存取的資源 (如每月的薪資表或是機密性的文件) , 就可能要改變 ASP.NET 的執行帳戶。這就有個問題了, 雖然可以透過設定 ASP.NET 的模擬帳戶 (Impersonate Account) 來提升 ASP.NET 的權限, 然而這卻不是個好作法, 原因只有一個－不要給 ASP.NET 應用程式過大的權限。

如果可以利用一個方法, 在需要存取較高權限的資源時, 利用暫時模擬 (Impersonation Temporary) 提高權限, 在存取完畢後, 立刻恢復到原本的權限, 如此就可以避免設定 ASP.NET 模擬帳戶的問題, 也可以存取到必要的資源。

這個可以透過 WindowsIdentity 類別的 Impersonate()方法來實作, Impersonate() 允許由程式來模擬特定的使用者帳戶, 它會回傳一個 Windows ImpersonationContext 物件, 若要將使用者還原時, 可利用它的 Undo()方法來復原先前的模擬狀態, 回到原本的使用者權限。

WindowsIdentity的建構式如果不接參數,預設是將程式模擬成WindowsIdentity本身指定的使用者。它也可以接一個參數 IntPtr, 這是一個指向整數資料的指標物件, 而這個整數資料是裝載著登入使用者的識別碼, 由 LogonUser() API 函式獲得。

LogonUser() API 由於不是.NET Framework 內建的函式, 要先宣告才可以使用：

```
[DllImport("advapi32.dll", SetLastError = true)]
public static extern bool LogonUser(
  String lpszUsername, // 使用者名稱
  String lpszDomain,    // 使用者所在網域名稱
  String lpszPassword,  // 使用者密碼
  int dwLogonType,      // 登入類型
  int dwLogonProvider,  // 登入驗證提供者
  ref IntPtr phToken);  // 登入帳戶識別元
```

LogonUser() API 可以透過不同的登入類型和不同的登入驗證提供者, 來實作不同的登入行爲, 通常是使用互動式登入 (LOGON32_LOGON_INTERACTIVE) 以及預設提供者(LOGON32_PROVIDER_DEFAULT) 來驗證。

接著, 來看看如何做吧。

解決方案

首先, 先將必要的 API 加入程式中。

```
[DllImport("advapi32.dll", SetLastError = true)]
public static extern bool LogonUser(String lpszUsername,
  String lpszDomain, String lpszPassword, int dwLogonType,
  int dwLogonProvider, ref IntPtr phToken);
```

```
[DllImport("kernel32.dll", CharSet = CharSet.Auto)]
public extern static bool CloseHandle(IntPtr handle);

[DllImport("advapi32.dll", CharSet = CharSet.Auto, SetLastError = true)]
public extern static bool DuplicateToken(
  IntPtr ExistingTokenHandle, int SECURITY_IMPERSONATION_LEVEL,
  ref IntPtr DuplicateTokenHandle);
```

接著, 在程式中先取得要模擬的使用者帳戶 (包含網域名稱、使用者名稱及密碼), 並呼叫 LogonUser() API 執行登入工作。

```
string userName, domainName, password;
// 取得 Domain, UserName 與 Password
domainName = this.T_DomainName.Text;
userName = this.T_UserName.Text;
password = this.T_Password.Text;

// 設定登入類型與提供者。
const int LOGON32_PROVIDER_DEFAULT = 0;
const int LOGON32_LOGON_INTERACTIVE = 2;
// 設定預設的 handle 爲零 (Win32 API 需要的)
tokenHandle = IntPtr.Zero;

// 執行 LogonUser() API
bool returnValue = LogonUser(userName, domainName, password,
  LOGON32_LOGON_INTERACTIVE, LOGON32_PROVIDER_DEFAULT,
  ref tokenHandle);
// 如果登入成功會回傳 true, 否則爲 false。
if (false == returnValue)
{
```

```
  // 登入失敗, 取得錯誤碼。
  int ret = Marshal.GetLastWin32Error();
  throw new System.ComponentModel.Win32Exception(ret);
}
```

取得使用者權杖識別碼 (Token ID) 後, 就可以利用它來執行模擬的工作。

```
// 使用已取得的使用者權杖, 產生 WindowsIdentity 物件。
WindowsIdentity newId = new WindowsIdentity(tokenHandle);
// 執行模擬。
WindowsImpersonationContext impersonatedUser = newId.Impersonate();
```

接著, 將傳回的 WindowsImpersonationContext 物件, 存到 Session 中, 以備工作結束後的還原, 或者, 只在函式中做模擬, 做完工作後立刻釋放 (這是最建議的作法) 。

```
// 將 WindowsImpersonationContext 物件存入 Session
Session["ImpersonateUser"] = impersonatedUser;
```

若要執行復原時, 將 WindowsImpersonationContext 物件由 Session 中取出, 然後呼叫它的 Undo() 方法, 即可將使用者權限復原到模擬之前的狀態。

```
// 取出 WindowsImpersonationContext。
WindowsImpersonationContext impersonatedUser =
  Session["ImpersonateUser"] as WindowsImpersonationContext;
// 復原模擬狀態。
impersonatedUser.Undo();
```

若要在程式中使用模擬方式來取得較高權限時, 切記一定要記得工作結束後要釋放掉, 否則會造成不安全的漏洞。

完整的程式碼如下。

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Runtime.InteropServices;
using System.Security;
using System.Security.Principal;
using System.Security.Permissions;

public partial class Part5_ImpersonateUser : System.Web.UI.Page
{
  [DllImport("advapi32.dll", SetLastError = true)]
  public static extern bool LogonUser(
    String lpszUsername, String lpszDomain, String lpszPassword,
    int dwLogonType, int dwLogonProvider, ref IntPtr phToken);

  [DllImport("kernel32.dll", CharSet = CharSet.Auto)]
  public extern static bool CloseHandle(IntPtr handle);

  [DllImport("advapi32.dll", CharSet = CharSet.Auto,
    SetLastError = true)]
  public extern static bool DuplicateToken(
    IntPtr ExistingTokenHandle, int SECURITY_IMPERSONATION_LEVEL,
    ref IntPtr DuplicateTokenHandle);
```

```
protected void Page_Load(object sender, EventArgs e)
{
  if (!Page.IsPostBack)
    this.labelUser.Text = WindowsIdentity.GetCurrent().Name;
}

protected void cmdImpersonate_Click(object sender, EventArgs e)
{
  IntPtr tokenHandle = new IntPtr(0);
  IntPtr dupeTokenHandle = new IntPtr(0);

  try
  {
    string userName, domainName, password;
    domainName = this.T_DomainName.Text;
    userName = this.T_UserName.Text;
    password = this.T_Password.Text;

    const int LOGON32_PROVIDER_DEFAULT = 0;
    //This parameter causes LogonUser to create a primary token.
    const int LOGON32_LOGON_INTERACTIVE = 2;

    tokenHandle = IntPtr.Zero;

    // Call LogonUser to obtain a handle to an access token.
    bool returnValue = LogonUser(userName, domainName, password,
      LOGON32_LOGON_INTERACTIVE, LOGON32_PROVIDER_DEFAULT,
      ref tokenHandle);

    Response.Write("LogonUser Called" + "<br>");
```

```
if (false == returnValue)
{
  int ret = Marshal.GetLastWin32Error();
  Response.Write(string.Format(
    "LogonUser failed with error code : {0}", ret));
  throw new System.ComponentModel.Win32Exception(ret);
}

Response.Write("Did LogonUser Succeed? " +
  (returnValue ? "Yes" : "No") + "<br>");
Response.Write("Value of Windows NT token: " +
  tokenHandle + "<br>");

// Check the identity.
Response.Write("Before impersonation: " +
  WindowsIdentity.GetCurrent().Name + "<br>");
// Use the token handle returned by LogonUser.
WindowsIdentity newId = new WindowsIdentity(tokenHandle);
WindowsImpersonationContext impersonatedUser =
  newId.Impersonate();

Session["ImpersonateUser"] = impersonatedUser;

// Check the identity.
this.labelUser.Text = WindowsIdentity.GetCurrent().Name;

// Free the tokens.
if (tokenHandle != IntPtr.Zero)
    CloseHandle(tokenHandle);
```

```
    }
    catch (Exception ex)
    {
        Console.WriteLine("Exception occurred. " + ex.Message);
    }
  }

  protected void cmdUndo_Click(object sender, EventArgs e)
  {
    WindowsImpersonationContext impersonatedUser =
      Session["ImpersonateUser"] as WindowsImpersonationContext;
    impersonatedUser.Undo();
    this.labelUser.Text = WindowsIdentity.GetCurrent().Name;
  }
}
```

此段程式碼是由.NET Framework 2.0 SDK 文件中的 WindowsIdentity.Impersonate() 方法的範例程式碼改寫而成, 原程式網址：

```
http://msdn2.microsoft.com/zh-tw/library/chf6fbt4(VS.80).aspx
```

相關問題

如果程式是來自檔案伺服器或非本機的來源, 可參考：

- Q54：我要如何在 ASP.NET 中存取網路磁碟 (或分享資料夾) 中的檔案？

Q68 如何使用驗證碼 (Authentication Code) 來防制大量重覆指令？

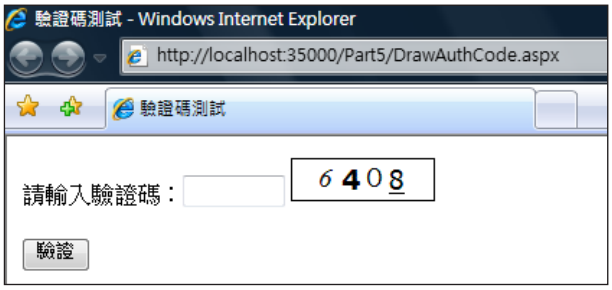
適用範圍：V ASP.NET 1.0 V ASP.NET 1.1 V ASP.NET 2.0 V ASP.NET 3.5

問題

我使用 ASP.NET 開發了一個會員管理系統, 初期運作還順暢, 但最近發現會員帳戶有大量增加的異常狀況, 所以我想在會員登錄註册的地方加上一道驗證的手續 (要求使用者輸入由系統產生的驗證碼) , 這個功能要如何做？

問題說明

筆者在「如何動態產生圖片」問答中, 提及了最近經常在網站系統中出現的驗證碼機制, 這是爲了防止類似網路機器人的防護功能。網站應用程式的註册功能如果過於簡單的話, 很容易被機器人的規則猜中, 而可能在很短的時間內被建立大量的帳戶, 而且帳戶名稱都會是有規則性的, 像是一些很流行的論壇系統 (phpBB, DotNetNuke 或是 vBulletin 等) 早期是沒有驗證碼機制的, 在後期才開始加上, 最主要的目的就是防止由機器人建立的垃圾帳戶。



驗證碼的設計應要考量幾個要點：

- 不可讓網站機器人有方法可以在輸入驗證碼前就獲得驗證碼, 表示驗證碼不要用明碼輸出到前端, 可用 Session 或加密後的 Cookie 來保護。
- 驗證碼圖片中的文字應該要有多樣的變化 (字型或字體) , 以及形狀的變化 (傾斜或倒立) , 其背景也可以有不同的顏色變化, 可防止具有圖形辨識能力的機器人藉由解析驗證碼圖片而取得驗證碼。

- 驗證碼不宜太長, 但也不宜太短, 4-8 個字元在可接受範圍, 驗證碼圖片所顯示出的字元雖要有變化, 但也要讓使用者可辨識。
- 若使用者輸入錯誤時, 驗證碼應變更成不同的字元與排列, 避免機器人使用字典法來破解。
- 可使用英文與數字字元組合或分散的排列法, 加強驗證碼的強度。

驗證碼圖片的產生方法, 可利用 HTTP Handler 來做, 請參考「Q25：如何動態產生圖片」問答。

解決方案

首先, 先建立 HTTP Handler, 並且將驗證碼產生的程式放在 HTTP Handler 中, 如程式 1。

程式 1：產生驗證碼圖片的 HTTP Handler

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Drawing;
using System.Drawing.Imaging;
using System.Web;

public class DrawAuthCode : IHttpHandler {

  public void ProcessRequest (HttpContext context)
  {
    // 經由 Cookie 取得驗證碼 (也可使用 Session)
    HttpCookie cookie = context.Request.Cookies["authCode"];
    string authCode = cookie.Value;
```

```
// 建立繪圖區, 繪圖區決定了驗證碼圖片的大小
MemoryStream ms = new MemoryStream();
Bitmap bmp = new Bitmap(100, 30);

// 建立繪圖物件
Graphics g = Graphics.FromImage(bmp);
List<Font> fontList = new List<Font>();
// 設定驗證碼要使用的字型
Font font1 = new Font("Times New Roman", 14, FontStyle.Italic);
Font font2 = new Font("Verdana", 14, FontStyle.Bold);
Font font3 = new Font("Bookman Old Style", 14,
  FontStyle.Regular);
Font font4 = new Font("Tahoma", 14, FontStyle.Underline);

// 隨機產生驗證碼所用的字型表 (本範例爲 4 碼)
while (fontList.Count < 4)
 {
  Random random = new Random();
  int fontIndex = random.Next(0, 5);

  switch (fontIndex)
   {
    case 1:
      if (!fontList.Contains(font1))
        fontList.Add(font1);
      break;
    case 2:
      if (!fontList.Contains(font2))
        fontList.Add(font2);
      break;
```

```
        case 3:
          if (!fontList.Contains(font3))
            fontList.Add(font3);
          break;
        case 4:
          if (!fontList.Contains(font4))
            fontList.Add(font4);
          break;
        default:
          break;
      }
    }

    // 執行繪圖
    g.FillRectangle(Brushes.White,
      new Rectangle(new Point(0, 0), bmp.Size));
    g.DrawRectangle(Pens.Black, new Rectangle(new Point(0, 0),
      new Size(bmp.Size.Width - 1, bmp.Size.Height - 1)));

    // 取得文字的大小 (含寬度與高度)
    SizeF sizeStr = g.MeasureString("1", new Font("Vendana", 14));

    // 計算文字置中的位置
    float drawY =
      (Convert.ToSingle(bmp.Size.Height) - sizeStr.Height) / 2;
    float drawX =
      (Convert.ToSingle(bmp.Size.Width) - sizeStr.Width * 4) / 2;

    g.DrawString(authCode[0].ToString(), fontList[0],
      System.Drawing.Brushes.Black, new PointF(drawX, drawY));
```

```
    g.DrawString(authCode[1].ToString(), fontList[1],
      System.Drawing.Brushes.Black, new PointF(drawX * 2, drawY));
    g.DrawString(authCode[2].ToString(), fontList[2],
      System.Drawing.Brushes.Black, new PointF(drawX * 3, drawY));
    g.DrawString(authCode[3].ToString(), fontList[3],
      System.Drawing.Brushes.Black, new PointF(drawX * 4, drawY));
    // 將繪圖完成的圖片資料寫入 MemoryStream 中
    bmp.Save(ms, ImageFormat.Gif);

    ms.Flush();

    g.Dispose();

    context.Response.ContentType = "image/gif";
    context.Response.BinaryWrite(ms.ToArray());

    bmp.Dispose();
    ms.Close();
  }
}
```

接著, 在需要驗證的頁面中, 產生驗證碼, 並寫入 Cookie 中, 讓 HTTP Handler 可以讀到驗證碼, 當然, 用 Session 也是可以的, 如程式 2。

程式 2：輸出驗證碼的 cookie

```
protected void Page_Load(object sender, EventArgs e)
{
  if (!Page.IsPostBack)
  {
```

```
    HttpCookie cookie = new HttpCookie("AuthCode");
    cookie.Expires = DateTime.Now.AddMinutes(5);
    // 使用隨機方式產生四碼驗證碼,
    // 若要更複雜的話可以使用英文字母和數字的組合
    cookie.Value = (new Random()).Next(1000, 10000).ToString();
    // 輸出 Cookie
    Response.Cookies.Add(cookie);
  }
}
```

有了驗證碼 Cookie 後, 就可以依照使用者所輸入的字元來做驗證, 如程式 3。

程式 3：執行驗證的程式

```
protected void cmdAuth_Click(object sender, EventArgs e)
{
  // 取出帶有驗證碼的 cookie
  HttpCookie cookie = Request.Cookies["AuthCode"];
  // 驗證使用者輸入的驗證碼是否正確
  if (cookie.Value == this.T_AuthCode.Text)
    Response.Write("驗證完成");
  else
  {
    Response.Write("驗證失敗");
    // 若驗證失敗時, 產生一組新的驗證碼
    cookie = new HttpCookie("AuthCode");
    cookie.Expires = DateTime.Now.AddMinutes(5);
    cookie.Value = (new Random()).Next(1000, 10000).ToString();
    Response.Cookies.Add(cookie);
  }
}
```

小常識 CAPTCHA

爲了防杜一些 SPAM Collector 或者是自動收集程式等工具對網站進行攻擊, 並確保進行資料登錄的是「人」而不是「程式」, 因此才有本問題所說的驗證碼方法, 而這個方法有更進一步的實作出現, 這是由 Luis von Ahn, Manuel Blum, Nicholas J. Hopper 與 John Langford 提出的CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) 機能, 它是以圖片輸出, 但將圖中的驗證碼做更多的處理, 像是歪斜、字型變化或是加上雜訊等功能, 讓電腦程式 (例如 OCR 光學辨識程式) 無法由這種圖片中自動辨識出驗證碼。

一個具有 CAPTCHA 能力的驗證碼的圖片的長相, 可參考這個網站：

```
http://www.cs.sfu.ca/~mori/research/gimpy/ez/
```

基於「有法固有破」的概念, 網路上也有出現一些專門解譯這種安全機能的程式, 但由於具 CAPTCHA 能力的驗證碼變化實在太多 (例如加上顏色或是背景擾亂等阻礙辨識的元素) , 故目前並沒有能夠破解所有 CAPTCHA 驗證碼的方法, 只有辨識程度的多寡而已。

而 CAPTCHA 的強化版本：reCAPTCHA 也已經問世, 由 reCAPTCHA 伺服器先產生 CAPTCHA 驗證碼圖片, 由前端網站抓取, 並且將使用者輸入的圖片傳送到 reCAPTCHA 的伺服器做檢查, 這個服務基本上是免費的, 可參考 reCAPTCHA Project 網站的說明。

- CAPTCHA的說明

```
http://en.wikipedia.org/wiki/Captcha
```

- reCAPTCHAProject

```
http://recaptcha.net/
```

Q69 如何簡化 URL 或是隱藏眞實 URL 以保護網站？

適用範圍： ✓ASP.NET 1.0 ✓ASP.NET 1.1 ✓ASP.NET 2.0 ✓ASP.NET 3.5

問題

我用 ASP.NET 開發了一個進銷存管理系統, 使用者可以利用我所設計的表單, 來查詢相關的產品資料, 但是公司提了一個需求, 說要讓使用者能直接在 URL 中就可以查詢, 請問我要如何做？

問題說明

隨著 Web 應用程式的複雜度愈來愈高, 開發人員在程式間傳遞資料的方法就會有所改變, 或者是使用多種的方法來一併處理, Query String 是其中一種, 可能只是要查個產品資料, Query String 就要寫的又臭又長：

```
ProductInfo.aspx?root=0&lv1=4&lv2=5&lv3=18&sp=134...
```

再者, 如一般的討論區論壇, 或是電子商務網站, 通常這樣的查詢字串會有可能洩露出產品在資料庫中的屬性, 而可能出現一些意料外的問題, 如果再加上 SQL Injection 之類的漏洞的話, 事情可能就很難處理, 難保會出現意外狀況 (駭客入侵) 。

而且如果使用者可以直接由 URL 就可以很方便的找到想要的頁面, 對於網站親和力來說也是可以加分的, 因此現在有很多的網站都開始在 URL 上下工夫了, 例如像 ASP.NET 的官方論壇, 就直接將討論串的 ID 放在 URL 中：

```
http://forums.asp.net/t/1187232.aspx
```

知名網路書店 Bookpool (經常有折扣的美國線上電腦書店) 也使用 URL 直接以 ISBN 就可以查到書籍：

```
Developing Drivers with the Microsoft Windows Driver Foundation
http://www.bookpool.com/sm/0735623740
Inside Microsoft Windows Communication Foundation
http://www.bookpool.com/sm/0735623066
```

或者是一些知名的 Blog, 其系統以 Blog 文章的標題來作爲 URL, 例如負責 Microsoft Learning 的 Trika, 他的 Blog 其中一篇文章的 URL 就是：

```
Trika's Blog:
http://blogs.msdn.com/trika
2007/6/27 - People at Microsoft Learning
http://blogs.msdn.com/trika/archive/2007/06/27/people-at-microsoft-learning.aspx
```

若想要實作出這樣具有彈性與互動性的 URL, 必須要由 Web 應用程式著手, 在輸入時將 URL 改變, 讓原本的 URL 仍然可以發揮作用, 這種功能稱爲 URL 重寫 (URL Rewriting)。

URL Rewriting 功能是一種攔截 URL 的機制, 將來自於 HTTP Request 的來源 URL 進行覆寫, 讓系統內部可以識別處理的一種特殊技巧, 這種技巧可以巧妙的將網站的結構和傳遞的資訊隱藏起來, 並且可以讓使用者輸入有別於以往的 URL 參數的方式, 呈現網站的資料。

URL Rewriting 可以做到不同的 URL 變化, 使用者可以輸入一些較具親和力的參數, 例如：

```
http://www.mybooks.com.tw/ProductInfo/Inside WCF
```

透過URL Rewriting, 可將它轉換成系統可看得懂的URL：

```
http://www.mybooks.com.tw/ProductInfo/ProductInfo.aspx?keyword=Inside WCF
```

或者是以階層的方法來表示的URL：

```
http://www.mybooks.com.tw/books/Microsoft Press/Windows/Introducing Win-
dows Server 2008
```

經由URL Rewriting, 可轉換成：

```
http://www.mybooks.com.tw/ProductInfo/ProductInfo.aspx?
type=books&vendor=Microsoft Press&category=Windows&caption=Introducing
Windows Server 2008
```

如此可以讓使用者直接在URL 中就下達指令, 而且透過URL Rewriting 的功能, 可以讓一些網站的結構和傳遞的參數得以隱藏起來, 以保護網站的結構及傳遞的參數類型而不被得知。

URL Rewriting在ASP.NET 多數是由HTTP Module來實作的(也可以用HTTP Handler, 但目前在網路上可看到的解決方法都是使用 HTTP Module 來做, 原因是HTTP Module 比HTTP Handler 更接近IIS, 而且不會受到副檔名的影響, 若只是要對一些特定的檔案做URL Rewriting, 則可考慮使用HTTP Handler來做) , 在應用程式開始處理來自於使用者的 Request 之前, 就將來源的URL 給重寫掉, 改成系統內部的參數, 再交由負責的程式來處理。

其實URL Rewriting的主要元件, 只有URL 重寫程式(Rewriting Engine)和URL 重寫規則(Rewriting Rules) , 重寫程式本身較不困難, 放在HTTP Module來處理即可, 但比較困難的就是要怎麼定義 Rewriting 的規則, 以及解析 (Parsing) 重寫的模式等, 因爲使用者可以輸入的規則千變萬化, 重寫規則可以定義較廣或較複雜的模式, 支援使用者的輸入規則, 或者是指示使用者, 以簡單的方式來輸入。

由於眞正具有全功能的 URL Rewriting 元件複雜度高且不易解釋, 因此筆者只使用較簡單的方法來展示 URL Rewriting 的作法, 若讀者對 URL Rewriting 有興趣時, 則可以參考一些網路上的作法。

解決方案

筆者展示的 URL Rewriting 功能, 是以北風資料庫爲基礎的資料, 可以允許使用者直接輸入產品名稱或客戶名稱來搜尋資料, 如果沒有給定, 就列出全部的資料, 例如：

```
// 查詢產品資料 (系統中其實並沒有 Products.aspx 這個檔案)
http://localhost/Products.aspx
// 查詢特定產品 (Ikura) 的資料
http://localhost/Products.aspx/ Ikura
// 查詢客戶資料 (系統中其實並沒有 Customers.aspx 這個檔案)
http://localhost/Customers.aspx
// 查詢特定客戶 (Chop-suey Chinese) 的資料
http://localhost/Customers.aspx/Chop-suey Chinese
```

首先, 在 App_Code 中產生一個類別程式檔案 (.cs) , 然後繼承 IHttpModule, 表示要撰寫 HTTP Module, 並建立必要的屬性和方法。

- ModuleName 屬性。
- Init()方法, 並設定一個 HttpApplication 參數。
- Dispose() 方法。

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Configuration;
```

```
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public class Rewriter : IHttpModule
{
  public Rewriter()
  {
    // empty constructor
  }

  public String ModuleName
  {
    get { return "RewriterModule"; }
  }

  public void Init(HttpApplication application)
  {
  }

  public void Dispose()
  {
  }
}
```

接著, 在 Init()方法中, 指示 HTTP Module 要處理 HttpApplication 的 BeginRequest事件。

```
public void Init(HttpApplication application)
{
  application.BeginRequest +=
    new EventHandler(this.Rewriter_BeginRequest);
}
```

然後, 將 URL 覆寫指令放到 BeginRequest 事件處理常式中。

```
private void Rewriter_BeginRequest(Object source, EventArgs e)
{
  HttpApplication application = (HttpApplication)source;
  HttpContext context = application.Context;

  // 分解 URL, 並取出參數
  if (context.Request.Url.ToString().IndexOf("Products") >= 0 ||
    context.Request.Url.ToString().IndexOf("Products/default.aspx")
      >= 0)
  {
    string[] productArgs =
      context.Request.Url.ToString().Split('/');
    // 將分解出的參數組合出對應的 URL,
    // 並呼叫 HttpContext.RewritePath()進行 URL 重寫
    if (productArgs.Length > 1)
      context.RewritePath("default.aspx?t=Products&name=" +
        productArgs[productArgs.Length - 1]);
    else
      context.RewritePath("default.aspx?t=Products");
  }
```

```
  else if (
    context.Request.Url.ToString().IndexOf("Customers") >= 0 ||
    context.Request.Url.ToString().IndexOf("Customers/default.aspx")
      >= 0)
  {
    string[] customerArgs =
      context.Request.Url.ToString().Split('/');
    // 將分解出的參數組合出對應的 URL,
    // 並呼叫 HttpContext.RewritePath()進行 URL 重寫
    if (customerArgs.Length > 1)
      context.RewritePath("default.aspx?t=Customers&name=" +
        customerArgs[customerArgs.Length - 1]);
    else
      context.RewritePath("default.aspx?t=Customers");
  }
}
```

其中的 HttpContext.RewritePath()在 .NET 1.x 中就已存在, 原本是給內建的 Cookieless 表單驗證使用的, 但現在已可以使用它來做 URL 重寫的工作。

由 HttpContext.RewritePath()重寫後的 URL, 會被傳遞到 default.aspx 中, 並且帶有參數, 此時 default.aspx 要有識別以及處理的程式。

```
protected void Page_Load(object sender, EventArgs e)
{
  if (Request.QueryString["t"] == null)
    return;

  SqlDataSource ds = new SqlDataSource();
  ds.ConnectionString = ConfigurationManager.ConnectionStrings[
    "northwindConnectionString"].ConnectionString;
```

```
  if (Request.QueryString["t"] == "Products")
  {
    if (!string.IsNullOrEmpty(Request.QueryString["name"]))
      ds.SelectCommand =
        "SELECT * FROM Products WHERE ProductName='" +
        Request.QueryString["name"] + "'";
    else
      ds.SelectCommand = "SELECT * FROM Products";

  }
  else if (Request.QueryString["t"] == "Customers")
  {
    if (!string.IsNullOrEmpty(Request.QueryString["name"]))
      ds.SelectCommand =
        "SELECT * FROM Customers WHERE CompanyName = '" +
        Request.QueryString["name"] + "'";
    else
      ds.SelectCommand = "SELECT * FROM Customers";
  }

  this.GridView1.DataSource =
    ds.Select(new DataSourceSelectArguments());
  this.GridView1.DataBind();

  ds.Dispose();
}
```

最後, 將這個 HTTP Module 註册到系統中, 在 Web.config 中加入這段指令：

```
<Configurations>
  <system.web>
    <httpModules> <!-- 註册 HTTP Module -->
      <add name="Rewriter" type="Rewriter"/>
    </httpModules>
  </system.web>
</Configurations>
```

至此, URL Rewriting 功能即算大功告成。

延伸閱讀

以下爲 URL Rewriting 的參考資料, 讀者若有興趣進一步研究時, 可參考這些文章, 或用 URL Rewriting 作關鍵字搜尋 Google 或 Windows Live。

- URL Rewriting in ASP.NET：

```
http://msdn2.microsoft.com/en-us/library/ms972974.aspx
```

- 15 Seconds : Rewrite.NET：

```
http://www.15seconds.com/issue/030522.htm
```

- URL Rewriting with ASP.NET：

```
http://www.codeproject.com/aspnet/URLRewriter.asp
```

- URLRewriting.NET:

```
http://www.urlrewriting.net/en/Default.aspx
```