なにはなくとも MIMEデコード

メールヘッダの解析と添付ファイルの取り出し

大澤 文孝 OSAWA, Fumitaka

はじめに

前稿で作成したPop3クラスでは、メールを1通ずつ取得して、Pop3Messageオブジェクトとして保存しました。

Pop3Messageオブジェクトでは、以下のように、ヘッダ部をheaderという変数名のNameValueCollectionオブジェクトに、ボディ部をbodyという名前のStringBufferオブジェクトに保存していました。

```
private NameValueCollection
  header = new NameValueCollection();
private StringBuilder
  body = new StringBuilder();
```

現時点では、headerやbodyは生のデータであり、人間が読むためには、デコードが必要です。

本稿では、メールのデコードの方法を説明します。

MIMEの基本

メールのフォーマットは、MIME（Multipurpose Internet Mail Extensions）として、RFC822やRFC2045～RFC2049で定義されています。

MIMEでは、

①ヘッダのエンコード方法
②ボディ部のエンコード方法
③添付ファイルの記述方法

などが定義されています。

図1に、ごく簡単なメールの例を示します。

これを基に、メールをデコードするには、どのような処理が必要なのかを説明していきます。

ヘッダのデコード

まずは、ヘッダ部分のデコードから説明します。

ヘッダに英数字以外を含む場合には、次の書式でエンコードしなければならないと定められています（RFC2047）。

=?文字コード?方式?文字列?=

日本語の場合には、「文字コード」の部分は、「ISO-2022-JP」であることが

レベル >>> Level
1 2 3 4 5

サンプル >>> Sample
この記事で取り上げたソースコードおよびサンプルプログラムは、
http://www.shoeisha.com/mag/windev/
からダウンロード可能です。

言語 >>> Language
- Visual C#

ツール >>> Tool
- Visual C# 2005 Professional

ほとんどです[注1]。

Quoted-PrintableとBase64

エンコード方式には、「Quoted-Printable」と「Base64」の2つがあります。

前者は「方式」の部分が「Q」、後者は「方式」の部分が「B」で示されます（図2）。

・Quoted-Printable方式

「=XX」という16進数の文字列で示す形式です。ただし「_」は空白（文字コード0x20）を示します。

元に戻すには、「XX」の部分をConvert.ToByteメソッドで変換します（リスト1）。

・Base64方式

この方式では、「8ビット×3文字」を「6ビット×4文字」へと変換します。

.NET Frameworkでは、Convert.FromBase64Stringメソッドを使うと、バイト列へと変換できます（リスト2）。

Quoted-Printable方式とBase64方式のどちらが使われるかは、「どの程度エンコードしなければならない文字列が含まれているか」によります。

というのは、Quoted-Printable方式では、「1文字が「=XX」と3文字」になるため、3倍の文字数が必要なのに対し、Base64方式では、3分の4倍になるだけで済むからです。

日本語の場合には、RFC1468の定めにより、Base64方式が使われることが

注1）ただし最近では、UTF-8が使われることも、まれにあります。UTF-8の場合には、「=?UTF-8?方式?文字列?=」となります。

図1：簡単なメールの例

```
Return-Path: <xxxxxxxx@xxxxx.xxx>
Received: from [127.0.0.1] (localhost [127.0.0.1])
  by xxx.xxxxx.xxx (8.12.8/8.12.8) with SMTP id k1F7Gn0t025256
  for <yamada@xxxxx.xxx>; Wed, 15 Feb 2006 16:16:50 +0900
MIME-Version: 1.0
Date: Wed, 15 Feb 2006 16:16:48 +0900
Subject: =?ISO-2022-JP?B?GyRCJWEhPCVrJE4lRiU5JUgbKEI=?=
From: xxxxxxxx@xxxxx.xxx
To: yamada@xxxxx.xxx
Message-ID: <JI2006021516164811.19416890@xxxxx.xxx>
Content-Type: text/plain; charset=ISO-2022-JP
Content-Transfer-Encoding: 7bit
X-Mailer: JsvMail 6.5 (Shuriken Pro4 /R.2)
X-Priority: 3
X-UIDL: YOB"!~h,!!([O!!$#9!!
Status: RO

このメールはテストです。
```

「ISO-2022-JP」で、「Base64エンコード」されていることを示す

ボディ部は「テキスト形式」で、「ISO-2022-JP」であることを示す。

ボディ部は「7bit」という形式で示されている（Quoted-PrintableやBase64は適用されていない）ということを示す。

Content-Typeヘッダにより、ボディ部は「ISO-2022-JP」で書かれているとのことなので、ユーザーにテキストとして表示するためには、「ISO-2022-JP」からの文字コード変換が必要。

図2：Quoted-Printable方式とBase64方式によるヘッダの表示例

・元の文字列

テストab漢字

・Quoted-Printable方式

=?ISO-2022-JP?Q?=1B$B%F%9%H=1B(Bab=1B$B4A;z=1B(B?=

文字コード　方式　エンコードした文字列（ISO-2022-JP）

・Base64方式

=?ISO-2022-JP?B?GyRCJUYlOSVIGyhCYWIbJEIOQTt6GyhC?=

文字コード　方式　エンコードした文字列（ISO-2022-JP）

リスト1：Quoted-Printableデコードする

```
static byte[] QuoteDecode(string src) {
  byte[] b = new byte[src.Length];
  int cnt = 0;
  for (int i = 0; i < src.Length; i++) {
    switch (src.Substring(i, 1)) {
      case "=":
        // 次の2文字を16進数で読む
        b[cnt] =
            Convert.ToByte(
            src.Substring(i + 1, 2), 16);
        i += 2;
        break;
      case "_":
        // 空白にする
        b[cnt] = 0x20;
        break;
      default:
        // そのままの値を採用
        Encoding.ASCII.GetBytes(src, i, 1, b,
                                cnt);
        break; }
    cnt++; }
  return b;
}
```

リスト2：Base64デコードする

```
static byte[] Base64Decode(string src) {
  // Base64デコードする
  return Convert.FromBase64String(src);
}
```