REALbasicでCGIを作る

田中求之 mact@antares.ecn.fpu.ac.jp

注意：

以下の文書は、技術評論社の雑誌「Macintosh Developer Journal」の私の連載「私のMacはWebサーバだ」のために書かれた原稿である（３２号に掲載）。原稿であるため、掲載された記事とは多少内容が異なる部分がある。また、記事の中でも述べているように、REALbasic　リリース１の正式版以前のバージョンを用いてあることには注意されたい。

この文書を個人が読み利用することに関してはなんらの制限は設けないが、文書（ファイル）の転載・再配布は堅く禁ずる。

１：はじめに

正式版のリリース以前の段階ですでに日本でも多くのユーザーを獲得しつつあるREAL Software社のREALbasic（以下、RBと略記）が、DR1r35でAppleEventを扱えるようになり、これによってRBによってCGIプログラムを作成することが可能になりました。RBを使えば、Threadを使ったマルチスレッド処理や、SocketによるTCP/IP通信を比較的簡単にプログラミングすることが可能ですので、この特徴を活かしたCGIプログラムを作成することができます。また、コンパイルによってアプリケーション化しますので、出来上がったCGIプログラムはAppleScriptのものに較べると高速で作動します。このように、RBがAppleEventをサポートしたことによって、CGIプログラムの開発環境として見たときも非常に魅力にあふれたものになったと言えるでしょう。そこで、今回はRBでCGIプログラムを作成する方法について、簡単に説明することにします。

なお、この記事の執筆のために用いたRBのバージョンはDR1r38です。執筆時点では、正式版はまだリリースされておらず、公開ベータテストによって精力的な改良とバグ取りが行われて、毎週のように新しいリリースが行われているという状況です[1]。また、AppleEventを扱うクラスやメソッドは実装されて間もないこともあって、不十分な点も見られ、完全に仕様が固まったとは言い難い状態にあります。ですから、皆さんがこれをお読みになるころ、あるいは正式に商品としてリリースされたものでは、この記事に書いたこととは異なっているかもしれません。その点にはご注意ください。

また、REALbasicそのものの紹介はしません。REALbasicについて知りたい方は、本誌２７号に土屋氏による「CrossBasicでのプログラミング」（REALbasicは、以前はCrossBasicという名前でリリースされていました）という紹介記事が載っていますので、そちらをご覧ください。

[1] http://www.realbasic.com/realbasic.html

２：MacOSのCGIプログラム

どのような環境で、またどんな開発言語を用いるにせよ、CGIプログラムを作成するにあたっては、そのプログラムを動かす環境（OS、サーバー、そして開発環境）においてCGIがどのように実装されているのかを理解しておくことが重要になります。MacOSのCGIについての詳しいことは、すでに本誌２５号の「あなたのMacでWebサーバを」第３回で解説してありますのでそちらを参照していただくとして[2]、最低限押さえておかなければならない事項を確認しておきます。

1) MacOSのCGIプログラムは、サーバーから送られてくるAppleEventを受け止めて作動するアプリケーションとして作成する必要がある。

2) CGIプログラムに送られてくるAppleEventは、Class='WWWΩ', ID='sdoc'のイベントである。アクセスに関する情報やFORMに記入されたデータなどは、このイベントのパラメーターとして送られてくる。

3) CGIプログラムは、処理が終了した時点で、サーバーから送られてきたイベントのreplyとして、何らかのデータを送り返す必要がある。この送り返されたデータは、特別なものを除いて、サーバーを経由して、ユーザーのブラウザにそのまま送られる。このため、送り返す（replyする）データの先頭には、必ず適切なHTTPヘッダを付けたHTTPプロトコルに則った形式のデータになっている必要がある。

4) FORMに入力されたデータは、METHODがGETの場合は、キーワード'kfor'のパラメーターに、METHODがPOSTの場合は、キーワード'post'のパラメーターに収められて送られてくる。いずれの場合も、データはURL形式にエンコードされているので、CGIプログラム側でデコードしなければならない。

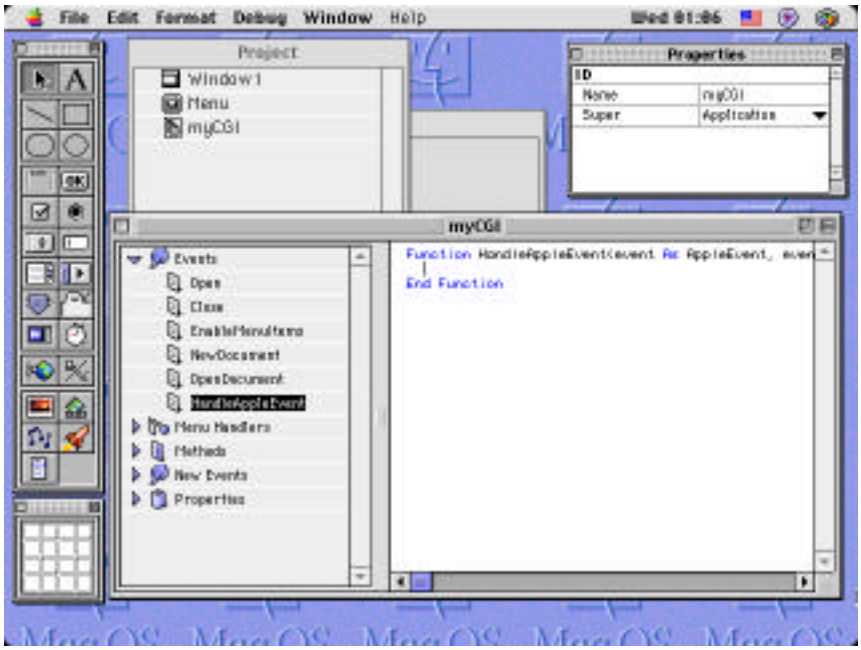
5) FORMのデータの形式は、フィールド名とそこに入力されたデータが=で連結され、さらに、フィールごとのデータを&で連結した形式になっている。このため、FORMからのデータを処理する際には、&をデリミタとしてデータをフィールドごとに切り分け、それぞれを更に=をデリミタとしてフィールド名とデータを切り離した上で、URL形式のデータをデコードするという手順を踏む必要がある。最初にURL形式のデコードを行ってしまうと、ユーザーによって入力された=や&によってデータの区切りを正しく捉えられない危険があるので、順序には気をつける必要がある

6) さらに、日本語のデータの場合は、漢字コードがJIS, SJIS, EUCの３つのいずれかの可能性があるため、URLデコードを終えたデータをもとに漢字コードの識別と変換を行って、自分の処理で用いる漢字コード（MacOSでは通常SJIS）に揃える必要がある。

MacOSのCGIプログラムを作成する場合、基本となるのは、サーバーから送られてきたAppleEventを受け止めて、処理をし、replyするアプリケーションを作るということです。RBにおいては、送られてきたAppleEventは、ApplicationクラスのHandleAppleEventイベントで処理することになりますので、Application　classをSuperとするClassを作り、このClassのHandleAppleEventイベントに、CGIのAppleEventの処理を実装し、これを中心としてCGIプログラムとしての処理を組み立てていくことになります。

---

[2] 記事をPDFにしたものを以下のURLで公開してあります
http://mtlab.ecn.fpu.ac.jp/mySamples/Understanding_CGI.pdf
http://mtlab.ecn.fpu.ac.jp/mySamples/Understanding_CGI.hqx



なお、FORMのデータのデコードや漢字コードの変換処理といったCGIプログラム作成に欠かせない機能もRBで組むことも可能だと思いますが、RBがXFCNやAppleScriptを取りこめることを利用して、この記事では、筆者が作成したXFCN[3]を利用することにします。

[3] http://mtlab.ecn.fpu.ac.jp/mySamples/XFCN_for_RB.hqx

３：REALbasicでのCGIプログラム

では具体的にCGIプログラムを作っていくことにしましょう。まず最初に、お決まりの(?)"Hello,World"の表示からはじめて、FORMページから送られてきたメッセージをファイルメーカーに登録するものまで、少しずつ機能を付け加えながら作ってみました。

●Hello.acgi

最初に、単に"Hello,　World"をページに表示するだけのCGIプログラムを作ってみます。プログラムとしては、まず、サーバから送られてくるAppleEvent（class='WWW ﾊ'、ID='sdoc'）を受け付けるようにし、そのイベントのreplyにHTTPヘッダとHTMLで書いた"Hello,　World"を送るというものになります。HTTPヘッダ部分の改行コードはCRLFが使われること、ヘッダとページ本体との間には必ず空行が必要なことなどに注意します。HandleAppleEventにリスト１のようなプログラムを書きます。

```
myCGI.HandleAppleEvent:
Function HandleAppleEvent(event As AppleEvent, eventClass As String, eventID As String)
As Boolean
  dim crlf, myReply as string

  crlf = chr(13) + chr(10)

  if eventClass = "WWWΩ" and eventID = "sdoc" then
    myReply = "HTTP/1.0 200 OK" + crlf + "Content-type: text/html" + crlf + crlf
    myReply = myReply + "<HTML>" + crlf
    myReply = myReply + "<HEAD>" + crlf
    myREply = myReply + "<TITLE>Hello, World</TITLE>" + crlf
    myReply = myReply + "</HEAD>" + crlf
    myReply = myReply + "<BODY>" + crlf
    myReply = myReply + "<H1 ALIGN=CENTER>Hello, World</H1>" + crlf
    myReply = myReply + "</BODY>" + crlf
    myReply = myReply + "</HTML>"

    event.ReplyString = myReply

    return true

  else
    return false
  end if
End Function
```

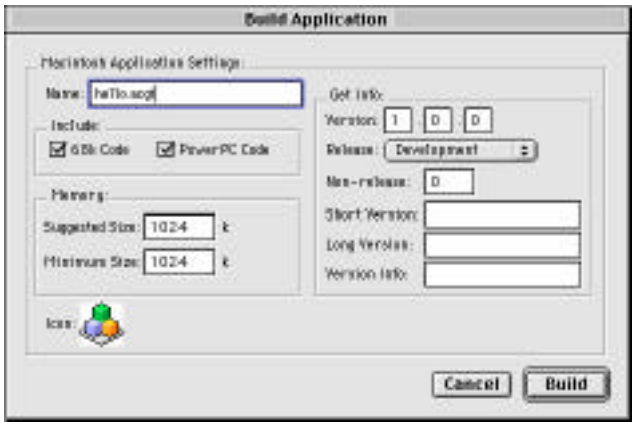
処理としては、単に"Hello. World"を表示するためのHTML（ページ）のデータを組み立てて、それをReplyしているだけの単純なものです。データの冒頭の部分にHTTPヘッダを付けていますが、HTMLで書かれたデータをブラウザに表示させたいときには、

```
HTTP/1.0 200 OK
Content-type: text/html
```

という２行のヘッダが最低限必要なヘッダとなります。この際、改行には必ずCRLFを使わなければなりません。また、ヘッダ部分の最後には必ず空行が１つ入ります。なお、HTTPヘッダに関しては、ここではこれ以上の説明は行いませんが、詳しく知りたい場合には、本誌３０号の「1998年のCGIプログラミング」をご覧ください[4]。

これを Build Application... によってアプリケーションして走らせるとCGIプログラムに

なります。なお、アプリケーションにする際には、かならず名前に拡張子.acgiを付けることを忘れないでください。ここではhello.acgiという名前でアプリケーションにすることにします。



ユーザーインターフェースは一切不要なのですが、そのままでは間抜けなので、とりあえずウィンドウにプログラムの名前でも書いておいてください。

このアプリケーションをWebサーバーと同じフォルダーに入れて走らせ

http://your.host/hello.acgi

にアクセスすると、図のように"Hello, World"と表示されます。

このCGIプログラムの中心であるHTMLのページを組み立てる処理をメソッド（リスト２のmakeHTMLメソッド）として切り離せば、HandleAppleEvent自体はリスト３のようなすっきりとしたプログラムになります。

[4] 記事をPDFにしたものを以下のURLで公開してあります
http://mtlab.ecn.fpu.ac.jp/mySamples/Understand_HTTP.pdf
http://mtlab.ecn.fpu.ac.jp/mySamples/Understand_HTTP.hqx

```
myCGI.makeHTML:
Function makeHTML(myTitle as string, myBody as string) As string
  dim crlf, myReply as string
  crlf = chr(13) + chr(10)

  myReply = "HTTP/1.0 200 OK" + crlf + "Content-type: text/html" + crlf + crlf
  myReply = myReply + "<HTML>" + crlf
  myReply = myReply + "<HEAD>" + crlf
  myREply = myReply + "<TITLE>" + myTitle + "</TITLE>" + crlf
  myReply = myReply + "</HEAD>" + crlf
  myReply = myReply + "<BODY>" + crlf
  myReply = myReply + myBody + crlf
  myReply = myReply + "</BODY>" + crlf
  myReply = myReply + "</HTML>"

  return myReply
End Function



myCGI.HandleAppleEvent:
Function HandleAppleEvent(event As AppleEvent, eventClass As String, eventID As String)
As Boolean
  dim myTitle, myBody as string

  if eventClass = "WWWΩ" and eventID = "sdoc" then
    myTitle = "Hello, World"
    myBody = "<H1 ALIGN=CENTER>Hello, World</H1>"
    event.ReplyString = makeHTML(myTitle,myBody)
    return true
  else
    return false
  end if
End Function
```

●Hello2.acgi

では、今度は、FORMからデータを受け取って処理するCGIプログラムを作ってみましょう。

まず最初に、FORMを使って名前を尋ね、その名前を使って"Hello,　XXXXX"という挨拶を返すという、先のサンプルを発展させたものを作ってみます。図のように表示されるリスト４のようなFORMから呼び出されるhello2.acgiというプログラムを作ることになります。

```
<FORM METHOD=GET ACTION="hello2.acgi">
お名前を教えてください：<INPUT TYPE=TEXT NAME=NAME SIZE=40>
<INPUT TYPE=SUBMIT VALUE="Send">
</FORM>
```

METHODがGETの場合、このFORMに記入されたデータは'kfor'パラメーターにString型（TEXT型）のデータとして収められてCGIプログラムに送られてきます。ですから、CGIプログラム側の手順としては、まずkforパラメータの値を取り出し、記入されたデータをURLデコードし、ページのデータを組み立てて、replyする、というものになります。

'kfor'パラメータを取り出すには、イベントのStringParamプロパティを使って

```
guestName = event.StringParam("kfor")
```

のようにすれば、読み出せます。

ユーザーが、たとえば「田中求之」と記入してSendをクリックした場合には、'kfor'パラメーターには次のような文字列が収まっています。

```
NAME=%93c%92%86%8B%81%94V
```

=の右側の部分が、URL形式にエンコードされた「田中求之」というデータです。ですから、この部分を取り出して、URLデコードを行えば、ユーザーの記入した名前がわかるわけです（漢字コードのことは、とりあえず考えないことにし、SJISで記入されてくるものとします）。

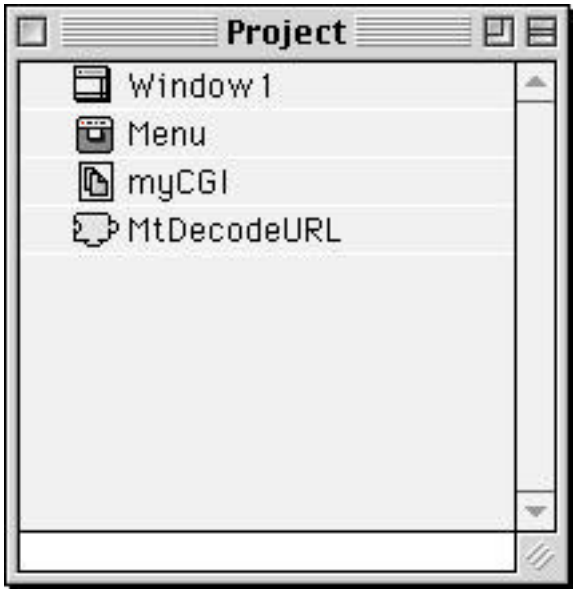
まず=の右側を取り出すには、NthFieldを使い

```
guestName = NthField(guestname,"=",2)
```

でいいわけです。

URLのデコードは、RBのMethodとして組むことも可能ですが、ここではXFCNを使うことにします（筆者がフリーウェアとして配付しているMtDecodeURL XFCN）。

MtDecodeURL XFCNのファイルをプロジェクトにDrag&Dropして図５のように登録しておけば、このXFCNを呼ぶだけでURLのデコードは完了です。あとは、これをもとにページを組み立てればよいことになります。



ということで、hello2.acgiのHandleAppleEventはリスト５のようになります。

```
myCGI.HandleAppleEvent:
Function HandleAppleEvent(event As AppleEvent, eventClass As String, eventID As String)
As Boolean
  dim guestName, myTitle, myBody as string

  if eventClass = "WWWΩ" and eventID = "sdoc" then

    guestName = event.StringParam("kfor")
    guestName = NthField(guestName, "=",2)
    guestName = MtDecodeURL(guestName)

    myTitle = "Hello, " + guestName

    myBody = "<H1 ALIGN=CENTER>Hello, " + guestName + "</H1>"

    event.ReplyString = makeHTML(myTitle,myBody)

    return true

  else
    return false
  end if
End Function
```

処理の結果はブラウザで図のように表示されます。

●SimpleForm.acgi

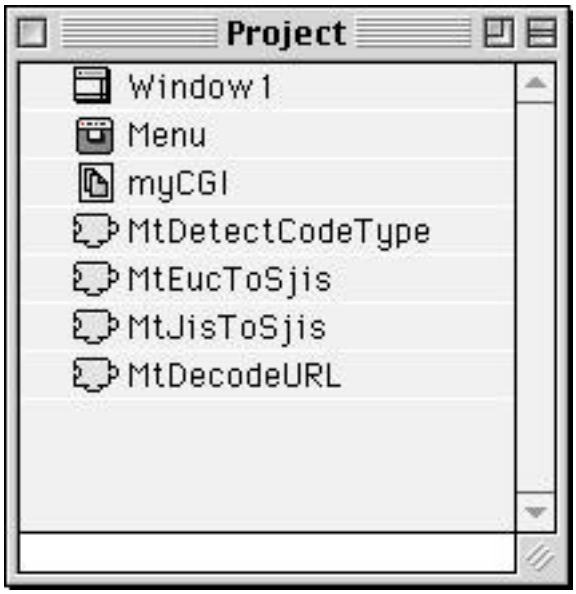
つづいて、FORMに記入され送られてきたメッセージをファイルに記録するCGIプログラムを作ってみます。

今度はFORMではMETHOD=POSTを用いることにします。この場合、FORMに記入されたデータはキーワード'post'のパラメーターによってCGIプログラムに送られてきます。処理の手順としては

1) postパラメーターを取り出す
2) 入力欄（フィールド）ごとに切り分ける
3) URLデコードを行う

4) 改行コードをCRに統一し、漢字コードの判定・変換を行う
5) ファイルに書き込む
6) サーバーへリプライを返す

というものになります。このうち、URLのデコードと漢字コードの処理はXFCNを使うことにします。



また、記録するファイルの名前は"Message.log"という名前でCGIプログラムと同じフォルダーに作成し、これに処理を行った日時と受け取ったメッセージを次々に追記していくことにします。そして、ユーザーには、簡単に"Thank you"というリプライを返すことにします。

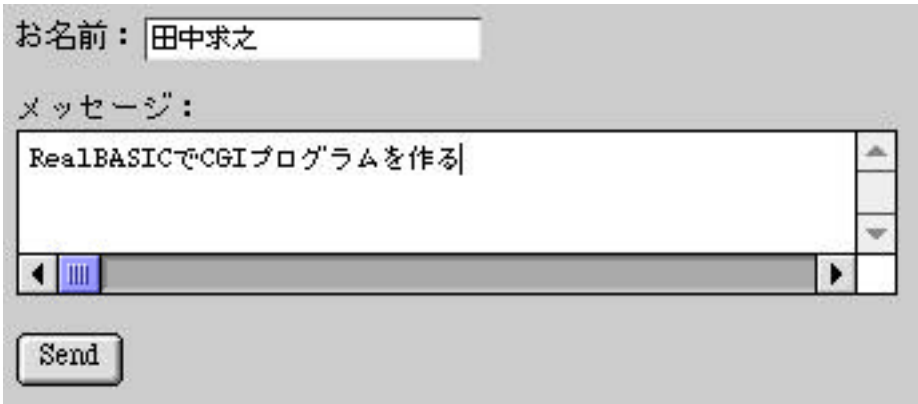
FORMのデータの処理については、すでに２５号の「あなたのMacでWebサーバを」などで説明したことではありますが[5]、CGIプログラミングの初心者が最初に引っ掛かる部分でもありますので、改めてRBに即して簡単に説明しておきます。

おおまかな手順としては、まずキーワードpostのパラメーターを取り出した後、最初にCountFieldで&で区切られたフィールドがいくつ存在するかを調べておき、あとはループでNthFieldを使ってそれぞれのフィールドを取り出し、さらにそれを、今度は=をデリミタとしてNthFieldを使ってフィールド名と入力されたデータとに切り分けていく、というものになります。これでは分かりにくいと思いますので、具体的に、データを示しながら説明します。

リストのようなFORMタグを書いたページに、図のようにユーザーが入力してSendボタンをクリックしたとします。

```
<TITLE>Test</TITLE>
<FORM METHOD=POST ACTION="/SimpleForm.acgi">
お名前：<INPUT TYPE=TEXT NAME=NAME><P>
メッセージ：<BR>
<TEXTAREA NAME=MSG COLS=50 ROWS=3></TEXTAREA><P>
<INPUT TYPE=SUBMIT VALUE="Send">
</FORM>
```

[5] 注2のドキュメントを参照



すると、CGIプログラムには、以下のようなpostパラメーターが渡されます（これをmyPostArgという変数で受け取ることにします）。

```
name=%93c%92%86%8B%81%94V&msg=RealBASIC%82%C5CGI%83v%83%8D%83O%83%89%83%
80%82%F0%8D%EC%82%E9
```

これを&をデリミタとして切り分けます。たとえば、最初の名前の入力データはNthField(myPostArgs,"&",1) で取り出せます。

```
temp = NthField(myPostArgs,"&",1)
```

これでtempは"name=%93c%92%86%8B%81%94V"になります。

次にこれをフィールド名とデータに切り分けるには、=をデリミタにしてNthFieldを使います。

```
NthField(temp,"=",1) --> "name"
NthField(temp,"=",2) --> "%93c%92%86%8B%81%94V"
```

そして、"%93c%92%86%8B%81%94V"をMtDecodeURLでデコードすれば"田中求之"という、ユーザーが入力した名前のデータが得られるのです。以下、同様に、各フィールドのデータをNthFieldを使って取り出していきながら、ログに記録するデータを生成します。そして、最後に漢字コードのチェックと必要であれば変換を行います。漢字コードの判定にあたっては、なるべく長い文字列で判定を行った方が精度があがりますので、ログ用のデータとして一つにまとめてから、処理を行うようにします。

漢字コードの変換、改行コードの統一、ファイルへの記録、そしてHTMLの組み立てはメソッドとしてHandleAppleEventとは独立させることにします。プログラムは次のようになります。

```
// CGIプログラムのメイン（AppleEventの処理）
myCGI.HandleAppleEvent:
Function HandleAppleEvent(event As AppleEvent, eventClass As String, eventID As String)
As Boolean
  dim myPostArg, myTitle, myBody as string
  dim temp, fName, myValue, myData as string
  dim fNum, i as integer

  if eventClass = "WWWΩ" and eventID = "sdoc" then

    myPostArg = event.StringParam("post")
    myData = ""

    //Parse post arguments

    fNum = CountFields(myPostArg,"&")
    For i = 1 to fNum
      temp = NthField(myPostArg,"&",i)
      fName = NthField(temp,"=",1)
      myValue = NthField(temp,"=",2)
      myValue = MtDecodeURL(myValue)

      myData = myData + fName + ":" + chr(13)
      myData = myData + lbToCr(myValue) + chr(13)
    next

    //Save to Log
    myData = changeToSJIS(myData)
    saveToLog myData

    //Reply
    myTitle = "Thank You!"
    myBody = "<H1>Thank your for your message</H1>"
    event.ReplyString = makeHTML(myTitle,myBody)

    return true
  else
    return false
  end if
End Function

//ファイルにデータを記録するメソッド
myCGI.saveToLog:
Sub saveToLog(myText as string)
  dim log as folderItem
  dim dx as TextOutputStream
  dim d as date
  dim myLog as string

  d = new date

  myLog = "** " + (d.shortDate) + " " + (d.longTime) + chr(13) + myText + chr(13)
  log = getfolderItem("Message.Log")

  dx = log.AppendToTextFile
  dx.writeLine myLog
  dx.close

End Sub
```

```
// 漢字コードをSJISに変換するメソッド
myCGI.changeToSJIS:
Function changeToSJIS(myText as string) As string
  dim myType as string

  myType = MtDetectCodeType(myText)

  if myType = "JIS" then
    return MtJisToSjis(myText)
  elseif myType = "EUC" then
    return MtEucToSjis(myText)
  else
    return myText
  end if

End Function

// 改行コードをCRに揃えるためのメソッド
myCGI.lbToCR:
Function lbToCR(myText as string) As String
  dim myData as string

  myData = ReplaceAll(myText,(chr(13)+chr(10)),chr(13))
  myData = ReplaceAll(myData,chr(10),chr(13))

  return myData

End Function


// リプライ用のHTMLを組み立てるメソッド
myCGI.makeHTML:
Function makeHTML(myTitle as string, myBody as string) As string
  dim crlf, myReply as string
  crlf = chr(13) + chr(10)

  myReply = "HTTP/1.0 200 OK" + crlf + "Content-type: text/html" + crlf + crlf
  myReply = myReply + "<HTML>" + crlf
  myReply = myReply + "<HEAD>" + crlf
  myREply = myReply + "<TITLE>" + myTitle + "</TITLE>" + crlf
  myReply = myReply + "</HEAD>" + crlf
  myReply = myReply + "<BODY>" + crlf
  myReply = myReply + myBody + crlf
  myReply = myReply + "</BODY>" + crlf
  myReply = myReply + "</HTML>"

  return myReply
End Function
```

これをSimpleForm.acgiという名前のアプリケーションとしてビルドし、Webサーバーと同じフォルダーの中にいれておけば、Formに入力されたデータを次々にMessage.logファイルに記録してくれるようになります。単純にフィールド名とそのデータを記録していくというものですから、よく言えばFORMの形式には依存してませんので、どんなFORMからのデータであっても使えます。

●msg2db.acgi　その１

サンプルの最後として、FORMに入力されたデータをファイルメーカーProのデータベースに登録するCGIプログラムを紹介します。現在ではファイルメーカーPro自体がWebサーバーになりますの

で、データベースの操作にわざわざCGIプログラムを作る必要はないのですが、ここではRBによるCGIプログラムと他のアプリケーションとの連携の実例として取り上げます。

図のようなCGI Messageというデータベースを作っておき、

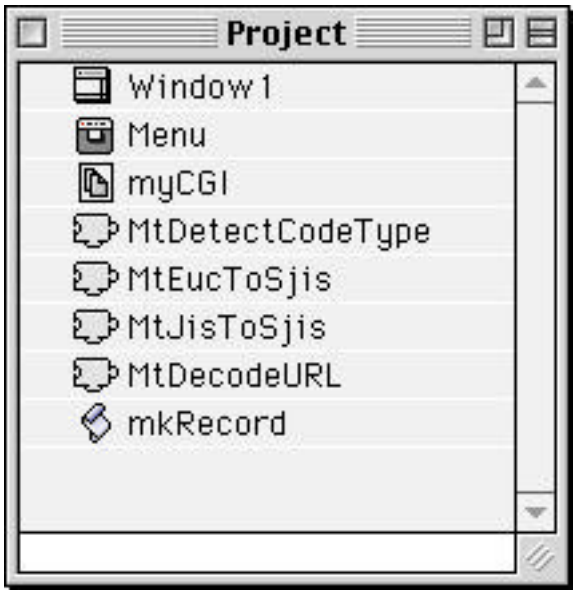
これに図のようなページから入力されたデータを登録していく、というプログラムにします。

RBからファイルメーカーのデータベースにデータを登録する場合、２通りの方法を用いることが可能です。一つ目の方法は、ファイルメーカーにデータを登録するAppleScriptのスクリプトを作成し、これを取り込んで利用するという方法です。もう一つは、必要なAppleEventをすべてRBで直接記述するという方法です。この両方について説明します。

まず最初に、AppleScriptを利用する方法から説明していきます。

RBでは、コンパイルしたAppleScriptを取り込んで利用することができます。この場合、スクリプトのrunハンドラーが呼び出されることになります。また、スクリプトに対してパラメーターを渡すことが可能ですが（文字列のみのようです）、その場合は、{}でパラメーターを並べたハンドラーにしておく必要があります。また、スクリプトから値を返す場合には、文字列を返すようにしておきます。ここではリストのようなスクリプトをmkRecordという名前でコンパイル済みのスクリプトとして保存し、これをDrag&Dropでプロジェクトに登録して使うことにします。

```
on run {gName, gAddr, gMsg}
    try
        tell application "ファイルメーカー Pro"
            create new record with data {gName, gAddr, gMsg} at database "CGI Message"
        end tell
        return "OK"
    on error
        return "Error"
    end try
end run
```



FORMからのメッセージをAppleEventのpostパラメーターから取り出して処理する部分は、先程のSimpleForm.acgiと同じですが、今回はセルごとの値を変数に割り当てるようにし、もし記入されてない欄があった場合には、記入ページ（FORMのページ）に差し戻すようにしましょう。また、処理後のリプライは、ページの内容を返すのではなく、ブラウザに対してしかるべきページにリダイレクトするように指示を返すことにします。この場合は、HTTPヘッダはステータスコード302とLocationヘッダで移動先を指示したものにする必要があります。処理後に表示するページは、データベース登録に成功した場合のページとエラーが生じた場合のページとを用意しておくようにし、それをTYPE=HIDDENのINPUTタグでFORMに書いておくことにします。

```
<HTML>
<HEAD>
<TITLE>メッセージをお寄せください</TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<H1>メッセージをお寄せください</H1>
<HR><P>
<FORM METHOD=POST ACTION="/msg2db.acgi">
お名前：<INPUT TYPE=TEXT NAME=NAME SIZE=40><P>
メールアドレス：<INPUT TYPE=TEXT NAME=ADDRESS SIZE=40><P>
メッセージ：<BR>
<TEXTAREA NAME=MSG ROWS=6 COLS=70></TEXTAREA><P>
<INPUT TYPE=HIDDEN NAME=OKPAGE VALUE="/thanks.html">
<INPUT TYPE=HIDDEN NAME=ERRPAGE VALUE="/DBerror.html">
<INPUT TYPE=SUBMIT VALUE="Send">
</FORM>
</BODY>
</HTML>
```

プログラムは以下のようになります。漢字コードの判定と変換の処理をメソッドとしてHandleAppleEventから独立させてあります。　また、先ほど述べたように、漢字コードの判定にはなるべく長い文字列を用いた方が精度が高まりますので、項目ごとのデータに切り分ける前に判定を行っておくようにしています。

```
myCGI.detectKcode:
Function detectKcode(myText as string) As string
  dim myData as string
  myData = MTDecodeURL(myText)
  return MTDetectCodeType(myData)
End Function

myCGI.changeToSjis:
Function changeToSjis(myText as string, myType as string) As String
  if myType = "JIS" then
    return MtJisToSjis(myText)
  elseif myType = "EUC" then
    return MtEucToSjis(myText)
  else
    return myText
  end if
End Function

myCGI.HandleAppleEvent:
Function HandleAppleEvent(event As AppleEvent, eventClass As String, eventID As String)
As Boolean
  dim myPostArg,okPage,errPage,replyMsg as string
  dim gstName,gstAddr, gstMsg, crlf as string
  dim myKtype as string

  if eventClass = "WWWΩ" and eventID = "sdoc" then

    crlf = chr(13) + chr(10)
    replyMsg = "HTTP/1.0 302 Found" + crlf + "Location: "

    myPostArg = event.StringParam("post")

    //Detect Kanji Code Type
    myKtype = detectKcode(myPostArg)

    //Parse post arguments
    gstName = NthField(myPostArg,"&",1)
    gstAddr = NthField(myPostArg,"&",2)
    gstMsg = NthField(myPostArg,"&",3)
    okPage= NthField(myPostArg,"&",4)
    errPage = NthField(myPostArg,"&",5)

    //Extract data
    gstName = NthField(gstName,"=",2)
    gstAddr = NthField(gstAddr,"=",2)
    gstMsg = NthField(gstMsg,"=",2)
    okPage = NthField(okPage,"=",2)
    errPage = NthField(errPage,"=",2)

    //Check Data
    if gstName = "" or gstAddr = "" or gstMsg = "" then
      replyMsg = replyMsg + "/form.html" + crlf + crlf
      event.ReplyString = replyMsg
      return true
    end if
```

```
    //Decode URL
    gstName = MtDecodeURL(gstName)
    gstAddr = MTDecodeURL(gstAddr)
    gstMsg = MtDecodeURL(gstMsg)
    okPage = MtDecodeURL(okPage)
    errPage = MTDecodeURL(errPage)

    //Change to SJIS
    gstName = changeToSjis(gstName,myKtype)
    gstMsg = changeToSjis(gstMsg,myKtype)

    //Line breaks to CR
    gstMsg = ReplaceAll(gstMsg,(chr(13)+chr(10)),chr(13))
    gstMsg = ReplaceAll(gstMsg,chr(10),chr(13))

    //crate new record
    if mkRecord(gstName,gstAddr,gstMsg) = "OK" then
      replyMsg = replyMsg + okPage + crlf + crlf
    else
      replyMsg = replyMsg + errPage + crlf + crlf
    end if

    event.ReplyString = replyMsg

    return true
  else
    return false
  end if
End Function
```

msg2db.acgiというアプリケーションにしてWebサーバのルートフォルダーに入れておき、ファイルメーカーでCGI Messageを開いておき、Formページにアクセスして入力を行うと、データベースにそれが書き込まれ、用意しておいたページが表示されるはずです。

●msg2db.acgi　その２

ファイルメーカーにデータを登録する処理を、AppleScriptを使わずに、すべてRBでプログラミングすることも可能です。先程のmkRecordの代わりにcreateRecordというメソッドを作り、そこにファイルメーカへのデータ登録の処理を組み込むことにしてみましょう。

この場合、AppleScriptで行っていたのと同じ処理に相当するAppleEventを、すべて自分でプログラムする必要があります。その際に重要になるのがAppleEventObjectSpecifierクラスの扱いです。

AppleEventのObject Specifierというのは、簡単に言ってしまうと、AppleEventの処理の対象となるオブジェクトの指示のことです。たとえば、先程のAppleScriptのスクリプトの中に

```
create new record with data {} at database "CGI Message"
```

というのがありましたが、これを実行すると、「データベース"CGI　Message"に、{}を内容とする、新しいレコードを、作成する」というAppleEventとしてファイルメーカーに送り付けられます。このAppleEventは、「作成する」という命令にあたるイベント（class=core,　ID=crel）に、「新しいレコードを」に相当する作成するオブジェクトの種類を示すパラメータ（keyword=kocl）、「{}を内容とする」に相当するオブジェクトの中身を指示するパラメーター（keyword=data）、そして「データベース"CGI Message"に」に相当するオブジェクトを作成する場所（これもオブジェクトです）を指示するパラメーター（keyword=insh）を伴ったものになります。この最後のオブジェクトの指示のパラメータに必要なデータ（どのオブジェクトにおいて処理

を行うかを指示するデータ）の型がObject Specifier Recordと呼ばれる、AppleEventでオブジェクトを指示するために定められた型のデータである必要があります。ようするに、RBでAppleEventを使って他のアプリケーションを操作しようとする場合には、「何をしたいのか」だけではなく、「どのオブジェクトを操作するのか」に関しても、すべて自分でAppleEventを組み立てる必要があるということです。

RBでは、このObject Specifier Recordを、AppleEventObjectSpecifierクラスとして扱えるようになっており、データを生成するメソッドも一通りそろっています。

実際にRBでAppleEventObjectSpecifierクラスを使うためには、Inside Macintoshの"Interapplication Communication" を読んでいObject Specifierを理解する必要がありますし、操作したいアプリケーションのオブジェクト・モデルの実装について調べる必要があります。オブジェクト・モデルの実装については、最終的にはAppleScriptで色々なスクリプトを動かしてみながら、どのようなAppleEventが用いられているのかをトレースしないと分からない場合もあります。このように、RBのAppleEventObjectSpecifierは、簡単には利用できないといったものですのが、それを使いこなすと、リスト型のデータを自在に扱えたり、実行速度が若干速くなったりといったメリットもあります。

説明が長くなりましたが、ここでは、先程のmkRecordスクリプトに相当するAppleEventをcreateRecordメソッドで組み立ててファイルメーカーに送り付けることにします。このメソッドはリストのようになります。このメソッドを、HandleAppleEventの中でmkrecordの代わりに呼ぶようにすればよいわけです（ただし、イベントがうまく送られたかどうかをbooleanで返しますので、プログラムをすこし変更する必要があります）。

```
myCGI.createRecord:
Function createRecord(gName as string, gAddr as string, gMsg as string) As boolean
  dim ev as AppleEvent
  dim myDesc as AppleEventDescList

  myDesc = New AppleEventDescList
  myDesc.AppendString gName
  myDesc.AppendString gAddr
  myDesc.AppendString gMsg

  ev = NewAppleEvent("core","crel","FMJ3")

  ev.MacTypeParam("kocl") = "crow"
  ev.DescListParam("data") = myDesc
  ev.ObjectSpecifierParam("insh") = GetNamedObjectDescriptor("cDB ",nil,"CGI Message")

  return ev.send

End Function
```

執筆時のDR1r38では、AppleEventの実装はまだ不十分なところが多く、特に送ったイベントのリプライを処理しようとしても限られた型のデータしか得られないという大きな問題があるため、AppleScriptで行えることをすべてRBのAppleEventの処理で行うことはできません。しかし、いずれはAppleEventを自由自在に扱えるようになるのではないかと期待しています。

４：おわりに

今回はREALbasicを使ってCGIプログラムを作成する方法について説明しました。AppleScript, MacPerlなどのスクリプティング環境で作ったものよりも高速で動くプログラムが作成でき、かつCやC++でプログラミングするよりは敷居は低くて簡単ですから、もしかしたら、MacOSのCGIプログラミングの主流になっていくかもしれません。

なお、今回は取り上げませんでしたが、RBを使ってAppleScript対応のアプリケーションを作成することも可能です（AppleEventの処理を実装し、aeteリソースをきちんと作っておけばよい）。たとえば、Socketによる通信機能を使ってメールサーバーと交信するAppleScript対応のアプリケーションを作っておき、それをAppleScriptのCGIプログラムの中で使っていくといった、おいしいところをうまく組み合わせたWebシステム構築が可能になったこと、これが根っからのAppleScript野郎である筆者には、一番うれしいことだったりします (^_^;;