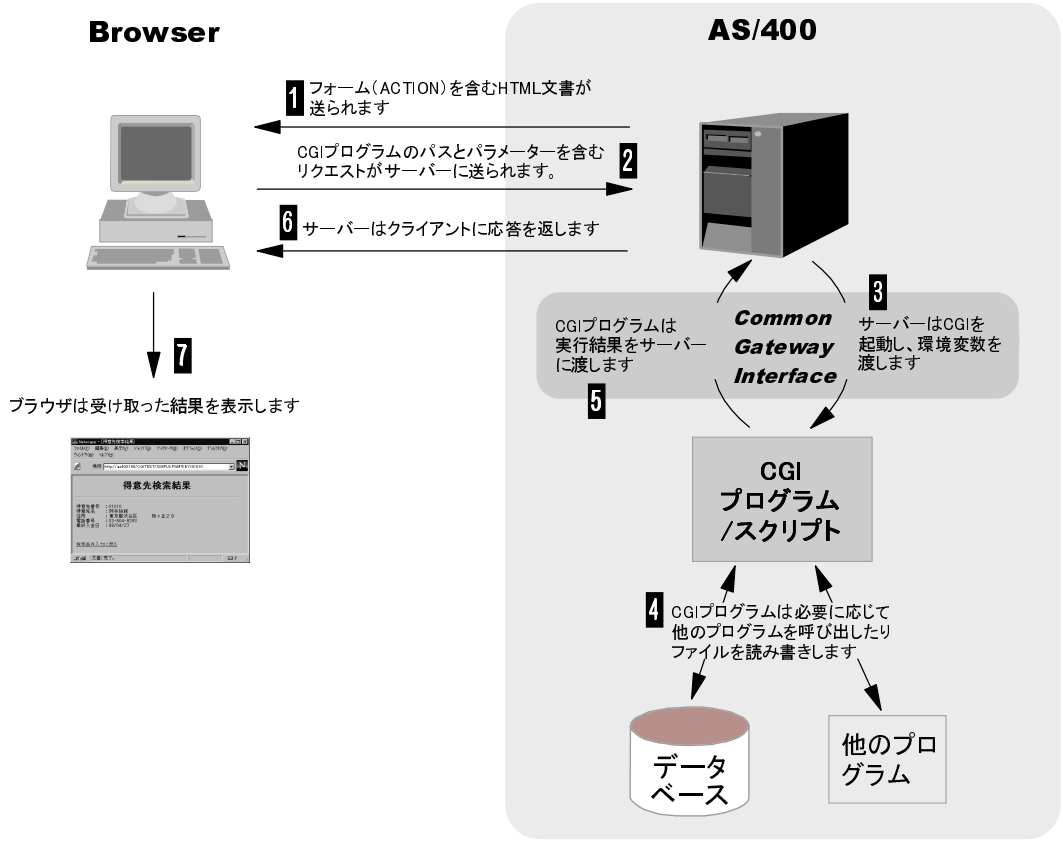
7 CGI

7.1　CGIプログラム



1 HTMLのフォームがサーバーからWebクライアントに送られます。フォームにはネットワーク上のCGIプログラムを指定するURLがACTIONとして記述されています。

2 ブラウザはURLをサーバーに送信します。URLにはCGIプログラムと(フォームで入力した)パラメーターが含まれています。

3 要求されたURLがプログラムであった場合、サーバーはそのプログラムを起動します。パラメーターはフォームで指定されたメソッド(POSTあるいはGET)に応じて標準入力か環境変数としてプログラムに渡されます。

4 サーバーに起動されたCGIプログラムは必要に応じて他のプログラムを実行したりファイル(データベースなど)を読み書きすることができます。

5 CGIプログラムはクライアントに返す情報を構築し、標準出力としてサーバーに返します。

6 サーバーはCGIプログラムから受け取った標準出力をブラウザに返します。通常はHTML形式で渡されます。

7 ブラウザはサーバーから受け取った情報をフォーマットし、表示します。

あなたのAS/400がWebサーバーとなり、IFS上のファイルをユーザーに提供することができるようになりました。美しい画像を含むHTML文書で掲示板を作ったり、社内文書を配布することもできます。さて、あなたはこれで満足ですか？　AS/400上のデータベースをWebブラウザからダイナミックに検索したいと思いませんか？　あるいは、ユーザーがWebブラウザ経由でデータを入力できたらと思いませんか？

このようなWebサーバーとWebブラウザの間での対話的な処理はCGI(Common Gateway Interface)を利用することによって実現できます。CGIはサーバーとサーバー上で動作するプログラム(あるいはスクリプト)とのインターフェースです。

HTMLは「フォーム」を開発することによってデータ(ユーザーの入力値など)をサーバーに返す機能を持ちます。これはユーザーインターフェースを単なるマウスクリックから照会やデータ入力に拡張できることを意味します。いくつものサイトがHTMLフォームと従来のアプリケーション間のインターフェースを書き、Webブラウザをそれらアプリケーションのフロントエンドにしています。これは開発者がクライアント側のコーディングに煩わされずクライアント／サーバー・アプリケーションを書ける可能性を開きます。

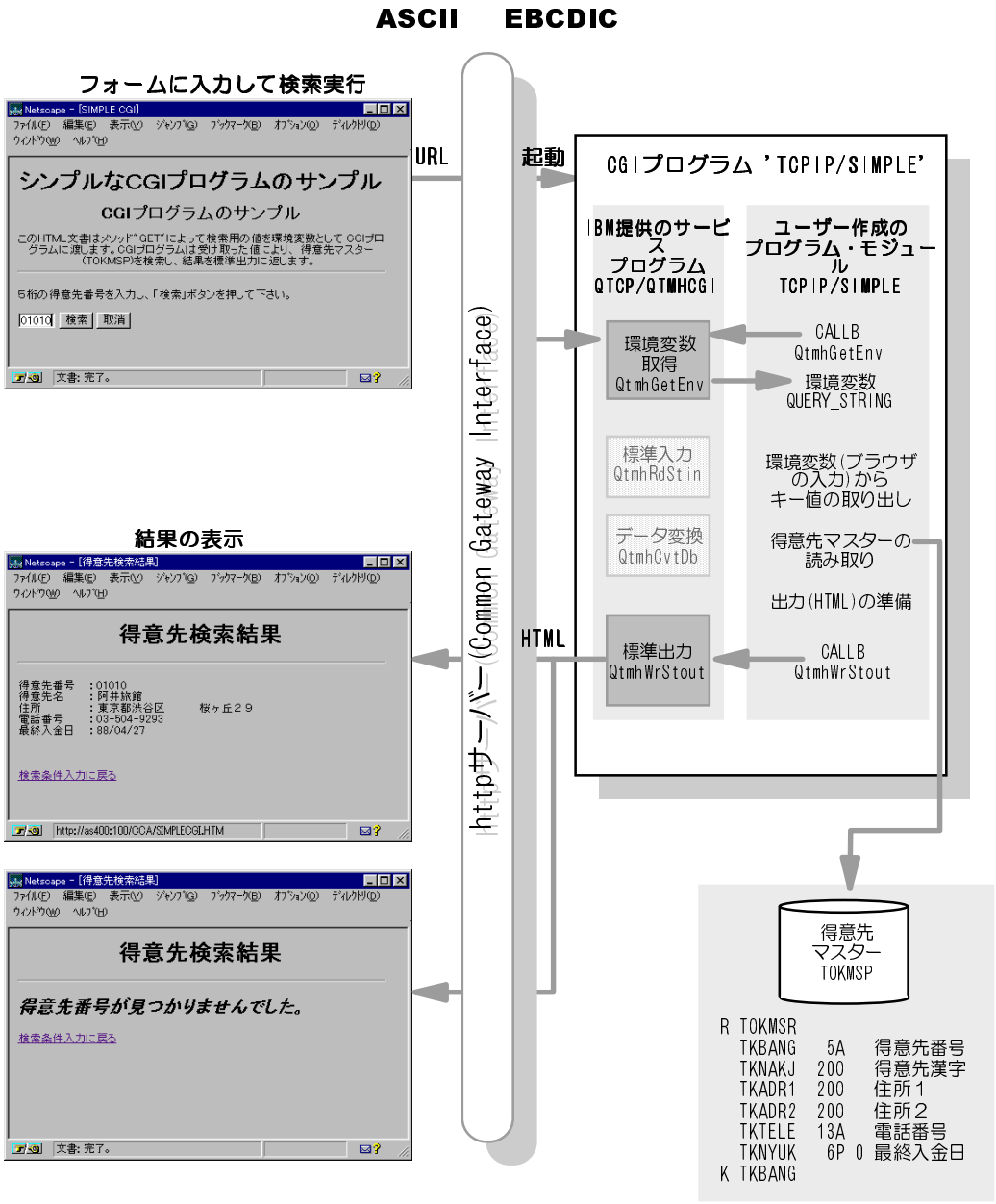
CGIにもHTTPと同様のバージョン番号があります。AS/400のCGIはバージョン1.1のサブセットであり、PUTやDELETEなどの処理は行えません。AS/400のCGIは次のようなサーバー機能をサポートします。

- プログラムの実行(CGI-BIN)
- サーバーサイド・クリッカブル・マップ
- DB2/400の利用(Net.Data)

7.2　CGIプログラムのサンプル

それでは実際に簡単なCGIプログラムを作成する過程を順を追って見ていきましょう。

このCGIプログラムはブラウザから5桁の得意先番号を受け取り、AS/400上の得意先マスターを検索し、検索結果をブラウザに返します。



7.2.1　フォームの作成

まず、フォームを含むHTML文書を作成します。

```
<HTML>
<HEAD><TITLE>SIMPLE CGI</TITLE></HEAD>
<BODY>
<CENTER>
<H1>シンプルなＣＧＩプログラムのサンプル</H1>
<H2>CGIプログラムのサンプル</H2>
<P>このHTML文書はメソッド"GET"によって検索用の値を環境変数として
CGIプログラムに渡します。CGIプログラムは受け取った値により、
得意先マスター(TOKMSP)を検索し、結果を標準出力に返します。
</CENTER>
<HR>
1 <FORM METHOD="GET" ACTION="/CGITEST/SIMPLE.PGM">
<P>５桁の得意先番号を入力し、「検索」ボタンを押して下さい。</P>
2 <INPUT TYPE=TEXT NAME="KEY" SIZE="6" MAXLENGTH="5" VALUE="">
3 <INPUT TYPE="SUBMIT" VALUE="検索">
<INPUT TYPE="RESET" VALUE="取消">
</FORM>
</BODY>
</HTML>
```

フォームを含むHTML文書「/CCA/SIMPLECGI.HTM」

1 入力フォームを表示します。<FORM>～</FORM>が一つのフォームとなります。フォームの中には<INPUT>や<SELECT>タグを使って「部品」を配置することができます。

2 <INPUT TYPE="...">はテキスト入力やボタンなどのフォーム部品を表示します。この場合はタイプが文字、フィールド名が「KEY」、部品の大きさ(テキストの場合は桁数)6、入力可能最大文字数は5、初期値に空値を指定しています。

3 実行ボタンを定義しています。「検索」という文字がついたボタンが表示され、このボタンをクリックすると<FORM>タグのACTIONで指定されたアクションが実行されます。

<FORM>タグでは「METHOD」と「ACTION」が指定されています。これらはそれぞれ入力値を「どのように」CGIプログラムに渡すか、「どの」CGIプログラムを起動するかを指定します。

「**METHOD**」はフォームに入力した値をCGIプログラム/スクリプトに渡す方式を指定します。METHOD=**GET**とすると、環境変数「QUERY_STRING」で、METHOD=**POST**とすると標準入力で渡すことになります。

> 環境変数や標準入力(出力)という概念はもともとUNIXで導入されたものです。AS/400ではCコンパイラを利用している場合を除いてなじみの薄いものでが、ここでは詳しい解説は割愛します。

一般的に入力長に制約の無いPOSTが使われることが多いようです。どちらの方法を使うかでCGIプログラムの入力値を受け取る部分のコーディングが変わります。この例では5桁の文字(入力されるのは数字のみ)が受け取れれば十分なので、扱いが簡単なGETを使うこととします。

「**ACTION**」は実行(SUBMIT)ボタンを押した時に実行するアプリケーションを指定します。ここにCGIプログラムのURLを指定すればそのプログラムがHTTPサーバーによって起動されます。他にも、ACTION="mailto:～"でメールが送信されますし、ACTION="JavaScript:～"でJavaScriptを実行することもできます。

さて、例ではACTION="/CGITEST/SIMPLE.PGM"となっていますが、このURLはAS/400上のどのプログラムを実行するのでしょう？ HTTPサーバーはブラウザから送られたURLを次のように解釈します。

HTML文書内のCGIを示すURL

```
<FORM METHOD=GET ACTION="/CGITEST/SIMPLE.PGM">
```

HTTPのEXECディレクティブ(WRKHTTPCFG)

```
EXEC /CGITEST/* /QSYS.LIB/TCPIP.LIB/*
```

AS/400で実行されるプログラム

```
/QSYS.LIB/TCPIP.LIB/SIMPLE.PGM
```

AS/400ではQSYS.LIBファイルシステム内のプログラムのみが実行可能なので上記の様にEXECディレクティブを指定します。「*」は総称名を表し、「その下の階層全て」を意味します。なお、EXECディレクティブが正しく実行されるようにディレクティブの順番には気を付けて下さい。例えば「PASS /*」が指定されたとすると、それ以降のMAPやEXECディレクティブは全て無効になります。

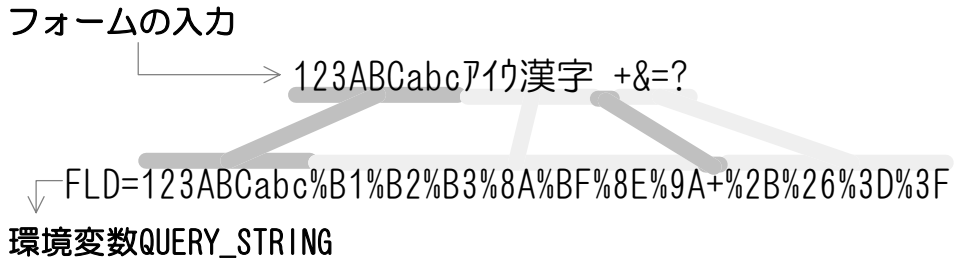
7.2.2 環境変数

通常、CGIプログラムはメソッドにかかわらず環境変数を参照して処理を行います。HTTPサーバーはCGIプログラムを呼び出す都度、環境変数を構築し直してCGIプログラムから利用可能にします。AS/400のHTTPサーバーが提供する環境変数には次のようなものがあります。

環境変数	内容	例
QUERY_STRING	フォームから入力された値	NAMAE=ABC&JUSHO=XXX
SERVER_NAME	サーバーのホスト名か、IPアドレスかDNS別名	as400.endicott.ibm.com
SERVER_PORT	要求が送られたポート名	80
REQUEST_METHOD	要求のメソッド。HTTPの場合はPOSTまたはGETのいずれか	GET
SCRIPT_NAME	実行されるプログラムのパス名	/CGITEST/SIMPLE.PGM
REMOTE_HOST	要求を発行したホスト名	royaha.roppongi.japan.ibm.com
REMOTE_ADDR	要求を発行したホストのIPアドレス	
CONTENT_TYPE	POSTのように追加情報を持つ場合はデータの内容の形式を表す	application/x-www-form-urlencoded
CONTENT_LENGTH	データの長さ(POSTの場合)	
IBM_CCSID_VALUE	サーバージョブのCCSID	

メソッドGETの場合、環境変数「QUERY_STRING」にフォームの入力値が入るので、CGIプログラムはこれを読み込んで処理を行います。QUERY_STRINGの内容は、フォームから入力された値そのままではなく、特殊な形式にエンコードされています。

Windows95のNetscape Navigator3.01[ja]とAS/400のHTTPサーバーの組み合わせでは、「FLD」という名前のテキスト入力フィールドにブラウザから「123ABCabcｱｲｳ漢字 +&=?」(「漢字」と「+」の間にスペースがあります)と入力してSUBMITすると、QUERY_STRINGは次のようにエンコードされます。



この場合のエンコードは次のルールで行われているようです。FC1630、1738にはこれ以外のエンコード・パターンの詳細も記載されています。

- フィールド名が付加され、値とは「=」で結ばれる複数のフィールドがある場合、「&」が区切り文字として使われる。
- 使用される文字は7ビットASCIIのみで、8ビットコード(SJISカタカナ、漢字のほとんどなど)、制御文字(CRLFなど)、特殊記号の多くは「%」の後に2桁の16進数文字を付けた形式(tripletと呼ばれる)で表す
- スペース(空白)は「+」に変換する

一方、メソッドPOSTの場合はフォームの入力値は標準入力として渡されるので、環境変数「QUERY_STRING」は空値です。CGIプログラムは最初に環境変数「CONTENT_LENGTH」を読み取り、その長さだけ標準入力からフォームの入力値を読み込みます。

CGIではブラウザの入力値は環境変数または標準入力としてCGIプログラムに渡されます。CGIプログラムの出力は標準出力としてCGIに渡さなくてはなりません。AS/400でCGIプログラムを作成できる言語として正式にサポートされているのはILE RPG、ILE COBOL、ILE Cの3つであり、これらの言語には必要な機能をサポートするAPIが用意されています。

APIのタイプ	ILE RPG	ILE COBOL	ILE C
環境変数の取得	QtmhGetEnv	←	getenv()
標準入力の読取	QtmhRdStin	←	fgets()など多数
標準出力への書出	QtmhWrStout	←	fwrite()など多数
CGI入力の解析	QtmhCvtDb	←	#pragma mapinc

Qtmhで始まるAPIのパラメーターの完全な解説はマニュアル「TCP/IP Configuration and Reference Version 3 SC41-3420-02」の「4.7.9 CGI Application Programming Interfaces」に記載されています。これらAPIを利用するにはCRTPGMコマンドのパラメーターBNDSRVPGMでサービスプログラムQTCP/QTMHCGIをバインドします。

7.2.3　ILE RPGによるCGIプログラム

各APIは英語大文字・小文字を区別するので、CRTSRCPFコマンドでソースファイル(通常QRPGLESRC)を作成する際に、パラメーターCCSIDで5035を指定します。5250エミュレーターも英小文字が使えるように設定(PC5250ではホストコードページとして「939 日本語英数小文字拡張」を指定)しておきます。

また、データベース「TOKMSP」がデフォルトのライブラリーリストに存在しない場合はCGIプログラムが実行時に使用するユーザープロフィール「QTMHHTP1」の現行ライブラリーに指定するか、RPGプログラムの中からOVRDBFコマンドで指定変更します。

```
     ****************  データの始め  ************************************
0001.00      * EOL/400 得意先マスター物理ファイル
0002.00     FTOKMSP    IF   E           K DISK
0003.00      * 環境変数取得 API 'QtmhGetEnv' のパラメーター
0004.00     DENBUFF           S             2048A   INZ
0005.00     DENBUFFLN         S                9B 0 INZ(2048)
0006.00     DENACTLN          S                9B 0 INZ
0007.00     DENVARNAME        S               20A   INZ('QUERY_STRING')
0008.00     DENVARLN          S                9B 0 INZ(12)
0009.00      * 標準出力書き出し API 'QtmhWrStout' のパラメーター
0010.00     DOUT              S             2048A   INZ
0011.00     DOUTLN            S                9B 0 INZ(2048)
0012.00      * コンパイル時配列
0013.00     DHTML             S                80    DIM(8) PERRCD(1) CTDATA
0014.00      * 作業変数（パック 6 桁→文字）
0015.00     DS                S                 6S 0 INZ(0)
0016.00      * EBCDIC New Line
0017.00     DNL               C                       X'15'
0018.00      * ユーザースペースエラーコード
0019.00      /COPY QSYSINC/QRPGLESRC,QUSEC
0020.00      *
0021.00      * 環境変数より検索パラメーター（得意先番号）を取り出します
0022.00   1 C                   CALLB     'QtmhGetEnv'
0023.00     C                   PARM                    ENBUFF
0024.00     C                   PARM                    ENBUFFLN
0025.00     C                   PARM                    ENACTLN
0026.00     C                   PARM                    ENVARNAME
0027.00     C                   PARM                    ENVARLN
0028.00     C                   PARM                    QUSEC
0029.00      *
0030.00     C                   MOVE      '00000'       KEY               5
0031.00   2 C                   EVAL      KEY = %SUBST(ENBUFF:5:5)
0032.00      * 得意先番号で得意先マスターを検索します
0033.00     C     KEY           CHAIN     TOKMSR                            90
0034.00      *   HTML のヘッダー
0035.00     C                   DO        4             I                 3 0
0036.00     C                   CAT       HTML(I):0     OUT
0037.00   3 C                   CAT       NL:0          OUT
0038.00     C                   ENDDO
0039.00      *   得意先が見つかった場合は内容を返します
0040.00     C     *IN90         IFEQ      '0'
0041.00     C                   Z-ADD     TKNYUK        S
0042.00     C                   MOVE      S             D                 6
```

RPGプログラムモジュール「TCPIP/SIMPLE」(1/2)

1 CALLBで環境変数取得API'QtmhGetEnv'を呼び出します。パラメーターENBUFFとENACTLNは戻り値でそれぞれ環境変数が保管されるバッファー(文字型変数)と環境変数の実際の文字列長を表します。その他のパラメーターはプログラム側で指定します。ENBUFFLNは環境変数が保管されるバッファーの長さ、ENVARNAMEは取得する環境変数の名前(ここでは'QUERY_STRING')、ENVARLNは取得する環境変数の名前の長さ(ここでは環境変数'QUERY_STRING'を検索するので12になります)を指定します。ENBUFFおよびENVARLNの変数の大きさは値を格納することができれば任意の長さで構いません。

2 例えばフォームで「01010」と入力すると、環境変数QUERY_STRINGには「KEY=01010」という値が入っているので、組み込み関数%SUBSTで必要な部分を取り出します

3 AS/400の仕様で、標準出力に書出すデータの各行は120バイト以下で、行の最後にEBCDICのNew Line(改行文字。16進数でX'15')が必要です。

```
0043.00      C                   EVAL      OUT = %TRIM(OUT) +
0044.00      C                             '<PRE>'                    + NL +
0045.00      C                             ' 得意先番号   : ' + TKBANG + NL +
0046.00      C                             ' 得意先名     : ' + TKNAKJ + NL +
0047.00      C                             ' 住所         : ' + TKADR1 +
0048.00      C                                                TKADR2 + NL +
0049.00      C                             ' 電話番号     : ' + TKTELE + NL +
0050.00      C                             ' 最終入金日   : ' +
0051.00      C                                          %SUBST(D:1:2) + '/' +
0052.00      C                                          %SUBST(D:3:2) + '/' +
0053.00      C                                          %SUBST(D:5:2) + NL  +
0054.00      C                             '</PRE><BR><BR>'           + NL
0055.00       *   得意先が見つからなかった場合はその旨表示します
0056.00      C                   ELSE
0057.00      C                   EVAL      OUT = %TRIM(OUT) + HTML(5) + NL +
0058.00      C                                   HTML(6) + NL
0059.00      C                   ENDIF
0060.00       *   HTML のトレーラー
0061.00      C                   EVAL      OUT = %TRIM(OUT) + HTML(7) + NL +
0062.00      C                                   HTML(8) + NL
0063.00       * 標準出力への書き出し
0064.00      C     ''            CHECKR    OUT           OUTLN
0065.00       *
0066.00      C                   CALLB     'QtmhWrStout'
0067.00      C                   PARM                    OUT
0068.00      C                   PARM                    OUTLN
0069.00      C                   PARM                    QUSEC
0070.00       *
0071.00      C                   SETON                                        LR
0072.00      C                   RETURN
0073.00       *
0074.00 **CTDATA HTML
0075.00 CONTENT-TYPE: TEXT/HTML
0076.00
0077.00 <HTML><HEAD><TITLE> 得意先検索結果 </TITLE></HEAD><BODY>
0078.00 <CENTER><H1> 得意先検索結果 </H1><HR></CENTER>
0079.00 <P><B><I><FONT SIZE=+2> 得意先番号が見つかりませんでした。
0080.00 </FONT></I></B></P>
0081.00 <A HREF="/CCA/SIMPLECGI.HTM"> 検索条件入力に戻る </a>
0082.00 </BODY></HTML>
        ****************  データの終り  ************************************
```

RPGプログラムモジュール「TCPIP/SIMPLE」(2/2)

このILE RPGプログラムはモジュールとしてコンパイルし、CRTPGMコマンドでサービスプログラム「QTCP/QTMHCGI」をバインドします。

```
1 > CRTRPGMOD MODULE(SIMPLE) DBGVIEW(*ALL)
    モジュール SIMPLE がライブラリー TCPIP に入れられた。最高の重大度は 00
      。 97/04/18 の 17:43:27 に作成されました。
  > CRTPGM PGM(SIMPLE) BNDSRVPGM(QTCP/QTMHCGI)
    プログラム SIMPLE がライブラリー TCPIP に作成された。
```

プログラムのコンパイル

1 後でこのCGIプログラムをデバッグする際にソースを表示できるようにDBGVIEWオプションを指定しています。

以上でCGIプログラムが完成しました。WebブラウザでHTML文書「SIMPLECGI.HTM」を開き、動作を確認します。

7.2.4　CGIプログラムのデバッグ

CGIプログラムをテスト／デバッグするにはいくつかの方法があります。

- 完成したプログラムをとりあえず実行し、ファイルQTEMP/QATMHSTOUTに書き出される内容(標準出力への内容)をチェック
- HTMLのフォームが無い場合、直接URLでCGIを呼び出せます。例えばこの例で得意先番号「03020」を指定した時の結果はブラウザのURLに「http://ホスト名/CGITEST/SIMPLE.PGM?KEY=03020」とすれば呼び出せます。
- TCP/IPの通信トレースを解析
- AS/400のデバッグ機能を利用

AS/400のデバッグ機能は次の手順で利用します。

AS/400のHTTPサーバージョブはWRKACTJOB JOB(QTH*)で表示することができます。いくつかのジョブの中で、状況が「DEQW」(待ち行列からの取出し操作の完了を待機中)のジョブがCGIを実行するジョブです。通常はこのようなジョブが複数個あり、どのジョブでCGIが実行されるかは不定です。あらかじめCHGHTTPAコマンドでパラメーターNBRSVR(2 2)としてジョブ数を1つにしておく必要があるでしょう。

```
                                活動ジョブの処理                          S100B86M
                                                                97/04/18  17:44:20
CPU %:     1.1      経過時間 :    07:15:15      活動ジョブ数 :    110

オプションを入力して，実行キーを押してください。
  2= 変更    3= 保留      4= 終了    5= 処理    6= 解放    7=ﾒｯｾｰｼﾞ の表示
  8=ｽﾌﾟｰﾙ･ﾌｧｲﾙ の処理    13= 切断 ...

OPT  ｻﾌﾞｼｽﾃﾑ/ｼﾞｮﾌﾞ   ﾕｰｻﾞｰ        ﾀｲﾌﾟ   CPU %    機能              状況
       QTHTT00385    QTMHHTTP     BCH        .0                     DEQW
       QTHTT16288    QTMHHTTP     BCH        .0                     SELW
```

ジョブが特定できたら、STRSRVJOBコマンドでサーバージョブに対してサービスジョブを開始し、STRDBGコマンドでCGIプログラムのデバッグを開始します。

```
> STRSRVJOB JOB(010804/QTMHHTTP/QTHTT00385)
> STRDBG PGM(SIMPLE) UPDPROD(*YES)
```

STRDBGコマンドを実行すると、コンパイル時にDBGVIEW(*ALL)オプションが指定されていればプログラムのソースが表示されます。

```
                        モジュール・ソースの表示

PROGRAM:   SIMPLE         LIBRARY:   TCPIP          MODULE:   SIMPLE
     1       * EOL/400 得意先マスター物理ファイル
     2      FTOKMSP    IF   E           K DISK
     3       * 環境変数取得 API 'QtmhGetEnv' のパラメーター
     4      DENBUFF           S            2048A   INZ
     5      DENBUFFLN         S               9B 0 INZ(2048)
     6      DENACTLN          S               9B 0 INZ
     7      DENVARNAME        S              64A   INZ
     8      DENVARLN          S               9B 0 INZ
     9       * 標準出力書き出し API 'QtmhWrStout' のパラメーター
    10      DOUT              S            2048A   INZ
    11      DOUTLN            S               9B 0 INZ(2048)
    12       * コンパイル時配列
    13      DHTML             S              80    DIM(8) PERRCD(1) CTDATA
    14       * 作業変数（パック 6 桁→文字）
    15      DS                S               6S 0 INZ(0)
                                                                  続く ...
デバッグ

F3= 終了プログラム    F6= 停止点の追加／消去    F10= ステップ    F11= 変数の表示
F12= 再開    F17= ウォッチ変数    F18= ウォッチの処理            F24= キーの続き
```

通常はこのあとブレーク・ポイント(停止点)を設定し、CGIプログラムを起動(ブラウザからフォームを実行)します。CGIプログラムが設定されたブレーク・ポイントに達すると画面が変わります。例えばカーソルをソース中の72行目に合わせてF6でブレーク・ポイントを設定した場合は次のようになります。

```
     69       * 標準出力への書き出し
     70      C                   CHECKR    OUT           OUTLN
     71       *
     72      C                   CALLB     'QtmhWrStout'
     73      C                   PARM                    OUT
     74      C                   PARM                    OUTLN
     75      C                   PARM                    QUSEC
                                                                  続く ...
デバッグ

F3= 終了プログラム    F6= 停止点の追加／消去    F10= ステップ    F11= 変数の表示
F12= 再開    F17= ウォッチ変数    F18= ウォッチの処理            F24= キーの続き
 回線 72 が停止点で停止した。
```

後はカーソルを変数に合わせてF11で変数の内容を表示したり、F10で一行ずつステップ実行したりしてプログラムの動作が意図したとおりであるかを確認します。

必要な事項を調べたら、忘れずにコマンドENDDBGおよびENDSRVJOBでデバッグを終了しておきましょう。

7.3　イメージマップ

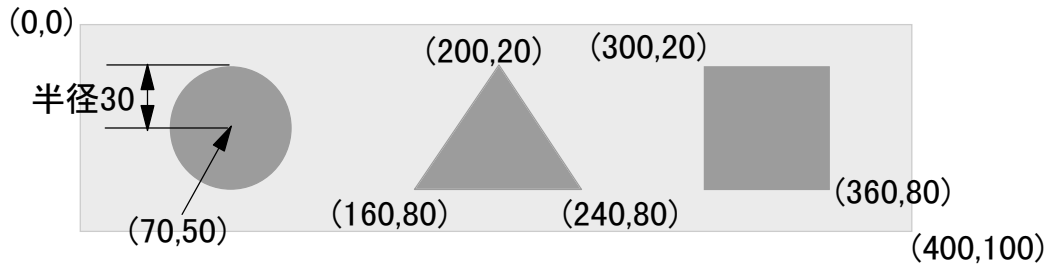
様々な公開されたホームページを見ていると、大きな画像の中で興味がある項目をマウスでクリックすると、その項目が記載されているページにジャンプするという構成のページが多いことに気が付くでしょう。これはイメージマップと呼ばれる機能によって実現されています。

イメージマップ(クリッカブルマップ)を利用すれば、画像の任意の点をマウスでクリックするだけで希望のリンクへジャンプすることができるようになります。イメージマップを実現するにはサーバーにURLで画像の座標だけを渡してサーバーがリンク先を判断する「サーバーサイド・イメージマップ」と、座標とリンクの判断をすべてブラウザ側で行う「クライアントサイド・イメージマップ」の2つの方法があります。

例として次のようなHTMLを使ってみましょう。

同じイメージが2つあり、それぞれ「サーバーサイド」、「クライアントサイド」のイメージマップ機能を利用しています。○、△、□、あるいは画像内のそれ以外の場所をクリックすると、同じ文書内の対応したタグにジャンプします。

例で使用しているイメージは、横400ドット×縦100ドットであり、それぞれの図形は次のような座標を持っています。



HTMLを次に示します。

```
<HTML>
<HEAD>
<TITLE>IMAGE MAP</TITLE>
</HEAD>
<BODY>
<A NAME="TOP">
<CENTER>
<H1>イメージマップのサンプル</H1>
<P>このHTML文書はサーバー及びクライアントのイメージマップをテストします。
下の図形の任意の点をクリックするとこの文書内の対応したタグにジャンプします。
</CENTER>
<HR>
<P>サーバーサイド・イメージマップ</P>
1 <A HREF="/CGIMAP/IMAGEMAP.MBR">
<IMG SRC="MAP.GIF" ISMAP></A>
<HR>
<P>クライアントサイド・イメージマップ</P>
2 <IMG SRC="MAP.GIF" USEMAP="#CLIENT" BORDER=0 WIDTH=400 HEIGHT=100>
<MAP NAME="CLIENT">
<AREA SHAPE=circle COORDS="70,50 30" HREF=#MARU>
<AREA SHAPE=polygon COORDS="160,80 200,20 240,80" HREF=#SANKAKU>
<AREA SHAPE=rect COORDS="300,20 360,80" HREF=#SHIKAKU>
<AREA SHAPE=default HREF=#SONOTA>
</MAP>
<P>
<HR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
3 <A NAME="MARU"><H2>○が選択されました。</H2><P><A HREF="#TOP">戻る</A>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<A NAME="SANKAKU"><H2>△が選択されました。</H2></P><A HREF="#TOP">戻る</A>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<A NAME="SHIKAKU"><H2>□が選択されました。</H2></P><A HREF="#TOP">戻る</A>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<A NAME="SONOTA"><H2>デフォルトが選択されました。</H2></P><A HREF="#TOP">戻る</A>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
<BR><BR><BR><BR><BR><BR><BR><BR><BR>
</P>
</BODY>
</HTML>
```

1 サーバーサイドイメージマップを指定しています。

2 クライアントサイドイメージマップを指定しています。

3 イメージマップをクリックした時のリンクを指定しています。この行は○に対応したタグ「MARU」が定義されています。

7.3.1　サーバーサイド・イメージマップ

サーバーサイド・イメージマップはIBM提供のCGIプログラム「QTCP/QTMHIMAG」によって実現されています。このプログラムはブラウザからURLで座標とリンク情報が記述されたソースファイル・メンバー名を受け取って処理を行います。

まず、HTTPのディレクティブでをいくつか追加する必要があります。ここでは次の3つのディレクティブを追加しました。既存の構成に追加する場合、他のPassやMapディレクティブと干渉しないように注意して下さい。

```
1 Map /CGIMAP/* /QSYS.LIB/QTCP.LIB/QTMHIMAG.PGM/QSYS.LIB/TCPIP.LIB/QTXTSRC.FILE/*
2 Exec  /QSYS.LIB/QTCP.LIB/*
3 Pass /QSYS.LIB/TCPIP.LIB/QTXTSRC.FILE/*
```

1 URLを短く(分かりやすく)します。
2 CGIプログラム「QTCP/QTMHIMAG」がAS/400のHTTPサーバーで実行できるようにします。
3 座標とリンク情報が記述されたソースファイル・メンバーにアクセス出来るようにします。

これらのディレクティブにより、HTMLの「アンカー」タグで指定されたイメージマップのURLは次のように解釈されます。

HTML文書内のイメージマップCGIを示すURL

```
<A HREF="/CGIMAP/IMAGEMAP.MBR><IMG SRC="MAP.GIF" ISMAP></A>
```

HTTPのMAPディレクティブ(WRKHTTPCFG)

```
MAP /CGIMAP/* /QSYS.LIB/QTCP.LIB/QTMHIMAG.PGM/QSYS.LIB/TCPIP.LIB/QTXTSRC.FILE/*
```

AS/400で実行されるプログラム

```
/QSYS.LIB/QTCP.LIB/QTMHIMAG.PGM
```

イメージマップのソース

```
/QSYS.LIB/TCPIP.LIB/QTXTSRC.FILE/IMAGEMAP.MBR
```

最後にイメージマップの座標とリンク情報を「形状　座標　リンク先」の順にソースファイルに記述します。なお、同じ領域に複数のイメージが指定された場合、先に指定された方が優先します。

```
        ****************  データの始め  ************************************
0001.00 CIRCLE (70,50) 30 /CCA/IMAGEMAP.HTM#MARU
0002.00 POLY (160,80) (200,20) (240,80) /CCA/IMAGEMAP.HTM#SANKAKU
0003.00 RECT (300,20) (360,80) /CCA/IMAGEMAP.HTM#SHIKAKU
0004.00 DEFAULT /CCA/IMAGEMAP.HTM#SONOTA
        ****************  データの終り  ************************************
```

ソースファイル・メンバー「TCPIP/QTXTSRC.IMAGEMAP」

7.3.2　クライアントサイド・イメージマップ

HTML3.0でブラウザによるイメージマップがサポートされました。これを利用すればサーバーに負荷をかけずにイメージマップを実現することができます。Netscape Navigator バージョン2以降、Microsoft Internet Explorer バージョン3以降などのブラウザでサポートされています。

前のサーバーサイド・イメージマップと同じ処理をするHTMLは次のように記述できます。同じ文書内のジャンプ先のタグを指定するには「#(タグ名)」と指定します。なお、Microsoft Internet Explorer 3.0は「SHAPE=default」をサポートしていないようですが、そのような場合は最後にrectでイメージ全体の座標とデフォルトのリンク先を指定します。

```
<IMG SRC="MAP.GIF" USEMAP="#CLIENT" BORDER=0 WIDTH=400 HEIGHT=100>
<MAP NAME="CLIENT">
<AREA SHAPE=circle COORDS="70,50 30" HREF=#MARU>
<AREA SHAPE=polygon COORDS="160,80 200,20 240,80" HREF=#SANKAKU>
<AREA SHAPE=rect COORDS="300,20 360,80" HREF=#SHIKAKU>
<AREA SHAPE=default HREF=#SONOTA>
</MAP>
```

クライアントサイド・イメージマップのHTML（抜粋）

クリッカブル・マップのそれぞれの方法の利点を考えてみます。

サーバーサイド・イメージマップ

- HTML3.0をサポートしない古いブラウザでも利用できる

クライアントサイド・イメージマップ

- サーバーに負荷をかけない
- イメージ上でマウスカーソルを動かすとリンク先のURLがブラウザのステイタス行に表示される
- 設定が比較的容易(HTMLのタグのみで実現)

いずれの方法を利用する場合でも、画像を表示できない/しないブラウザからの利用を考慮し、同じリンク情報を文字でも記載しておくといった気配りが必要でしょう。