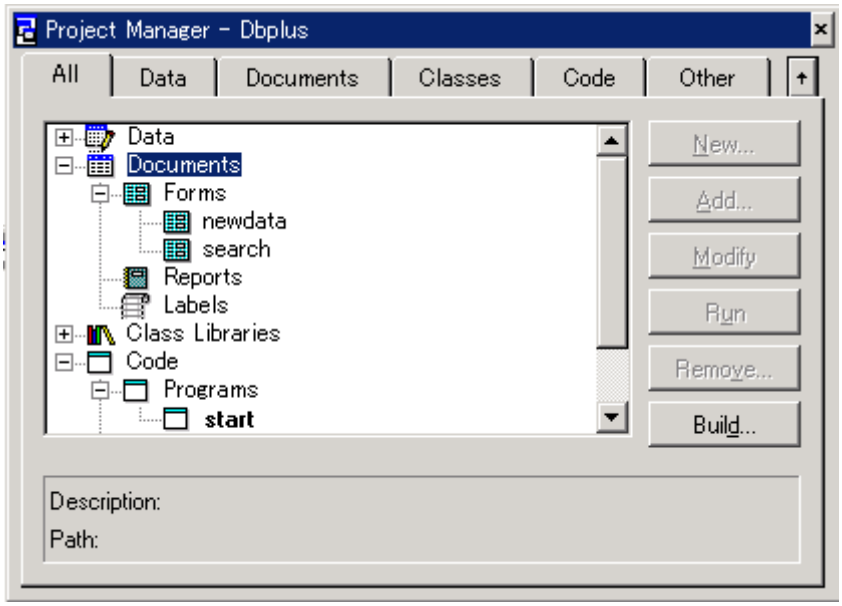
VisualFoxPro を触りだしてから約半年が経過し、やっと動くアプリケーションが作れるようになった。実はこの一月で多くの事を学びこれをお伝えするだけで数ヶ月掛かりそうだなと思っていた所に、本誌の全面刷新の話が飛び込んできた。実は個人的にも情報誌としての雑誌の時代は終わったと思っていたので、ある意味では良かったという感想を持っている。インターネットがここまで普及すると情報の量とか早さを売りにする雑誌を読んで助かるのは、本当に一部の層に限られてくるだろう。だから雑誌のターゲットを変えてもきっと結果は変わらないので、今度は雑誌のポリシーそのものを変えなくてはならないだろうと思っている。例えば最近のベストセラーである「チーズは何処に消えた」は「CHANGE TO DIE」の話で、変わる事を恐れていた主人公が、変わる事を恐れなくなって幸せになる話である。同様に「金持父さん、貧乏父さん」も勝つためにヒケツであって、両方の本は少しエゲツナイと感じるかもしれないが、情報を集めた本ではない。今年になってからオフィスの本を数千冊捨てたのだが、真っ先に捨てたのは情報を集めた雑誌とか本であった。これらは全く不要なのである。連載を最後にして筆者自身の考え方を述べて欲しいという編集部からのリクエストがある。これについては本稿の最後に書かせていただくとして、VisualFoxPro の総集編をお届けしよう。

「EXE プログラムを作る」

データベースといえば MS の ACCESS しか知らないという読者の方も多いと思うので、ACCESS と VisualFoxPro は何処が違うのか？を最初にご説明しよう。ACCESS は専門的にはデータベースアプリケーション、VisualFoxPro はコンピュータ言語に分類される。この二つを使ってそっくりなアプリケーションを作る事が可能ではあるが、その二つの違いは ACCESS で作った物は ACCESS 自体がないと動かないが、VisualFoxPro で作られたものは VisualFoxPro がなくても動く。だから VisualFoxPro で作られたものは自由に配布できるが、ACCESS で作られたものは ACCESS を持っている相手だけにしか配布出来ない。また新しいバージョンの ACCESS で作られたプログラムは古い ACCESS では動かない。VisualFoxPro で作られたものはユーザにはそのプログラムコードは全く見えないし、修正も出来ないが、ACCESS ではコードや内容が見えてしまう。ちょっと知っているユーザがいると、ACCESS で作られたものは変更されてしまう可能性がある。ACCESS と VisualFoxPro はまだ大きな差があるのだが、それはもう少し後でご説明するとして非常にすっきりと VisualFoxPro のアプリケーションを構築する方法に至ったので、今回はこれをご説明しよう。最初に画面１をご覧頂きたい。これは既に完成した VisualFoxPro のアプリケーションのプロジェクトマネージャーである。VisualFoxPro でアプリケーションを構築する場合は必ずプロジェクトマネージャーを利用する。VisualFoxPro でプログラムを作る場合は最初にフォームから作り始めるのだが、画面１の下側に CODE→PROGRAMS→START とあるここを最初に作っておく事がポイントである。START という名前は何でも良い。これが最初に動く部分という意味である。このプログラムはプログラムマネージャーで New ボタンを押して新規に作られるか、Add で既にあるものを取り込むかのどちらか

で用意される。修正は Modify で行う。実際のコードを見てフォームを含め、他のコンポーネントとの関係を調べてみよう。プログラム１にコードを示す。

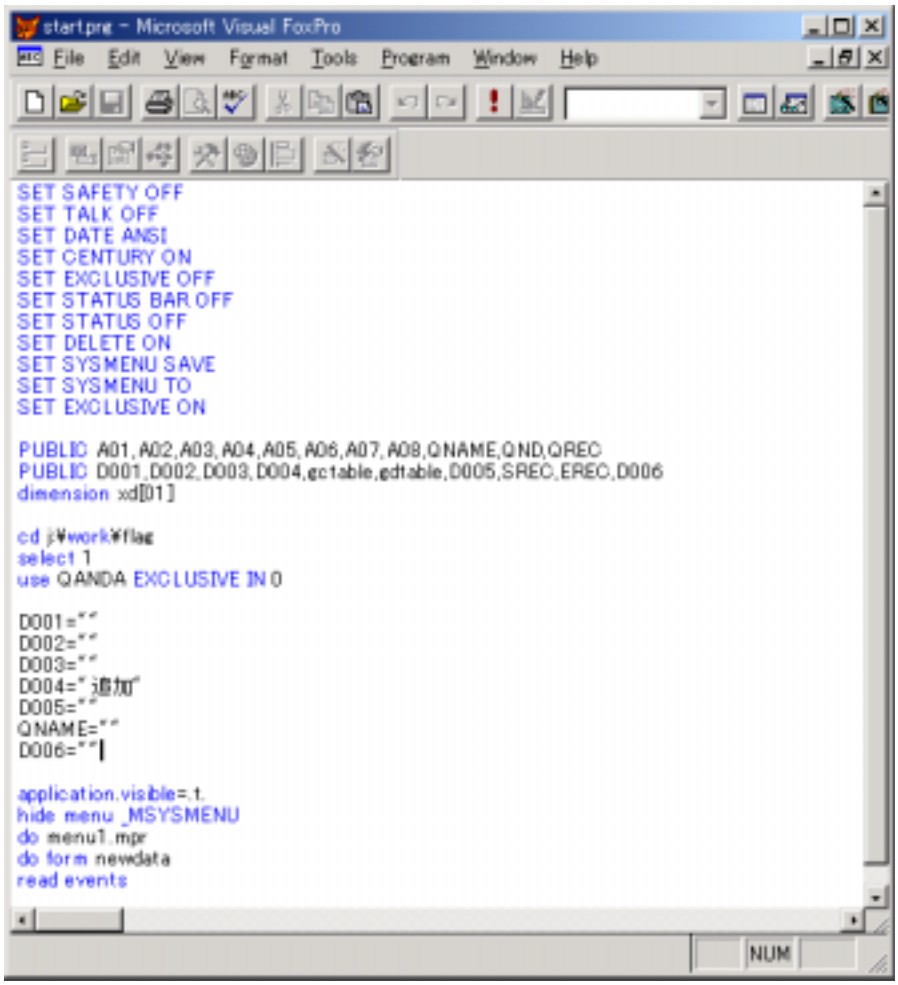


画面１プロジェクトマネージャー

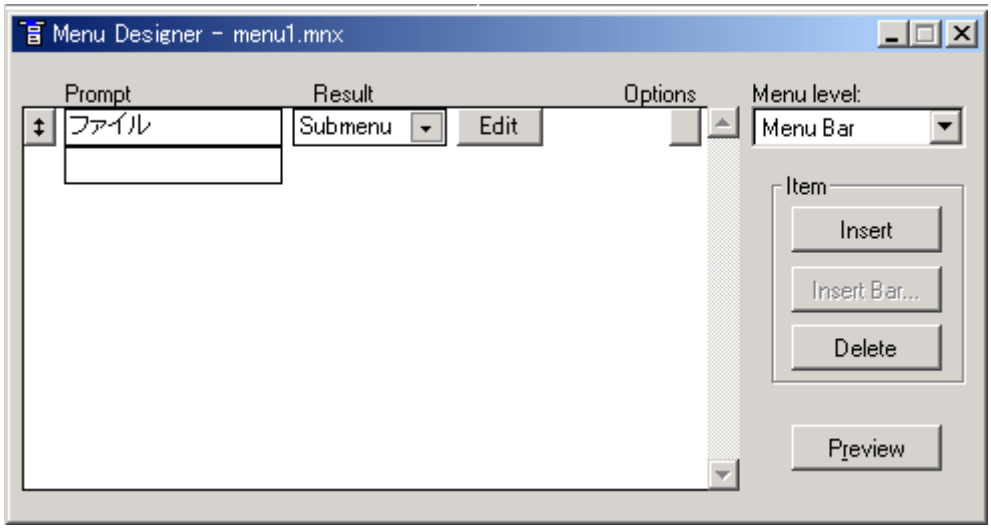
さてコードの内容を見てすぐに理解されるのは、古くからの Xbase 言語のコードであり、特に目新しい部分は最後の方まで出てこない事だろう。VisualFoxPro らしい部分は次の４行であり、それぞれは以下の意味を持っている。

hide menu _MSYSMENU	&&	システムメニューを隠す
do menu1.mpr	&&	メニューの読み込み
do form newdata	&&	フォームを表示
read events	&&	ここまでの実行

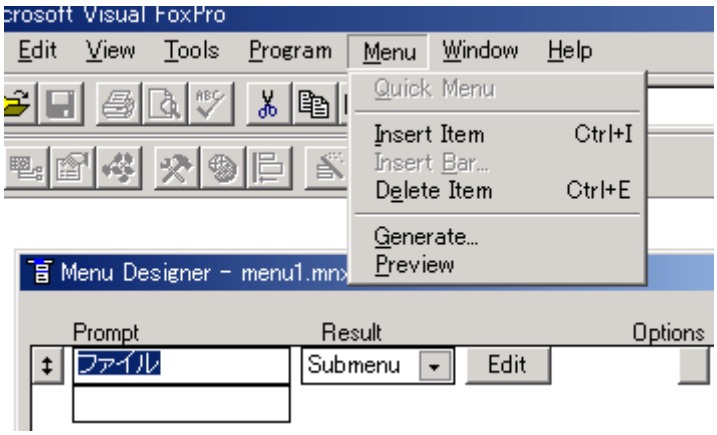
既にフォームには二つのフォーム（newdata と search）が登録されているが start に記述するフォームは親フォームである。親フォームから子フォーム（search）を呼び出すので、子フォームの記述は不要となる。メニューはやはりプロジェクトマネージャーの other→menu から作成する。画面２ではメニューを実際に作成している様子である。メニューの作成自体は実際に動かした方が理解しやすいので細部は割愛するが、このメニューをプログラムで動くようにする為には、拡張子が mpr のファイルとして生成する必要がある。（画面３）ここで生成したメニューの一つを親ウィンドウに置くには、プログラムの中で do menu1.mpr と記述する。メニューを操作して起こすアクションは全てメニューを作成する画面２の中での作業となる。メニューにはプルダウンとポップアップの二つが選べるようになっていて、他で作ったメニューファイルや機能を再利用する事も可能である。フォームの作成は WEB アプリケーションを作成する場合と何も変わらないので特に説明は不要だと思うが、親フォームと子フォームの関係で重要になるのはフォームのモードである。画面４は説明に使っているプロジェクトから生成されたアプリケーションであるが、



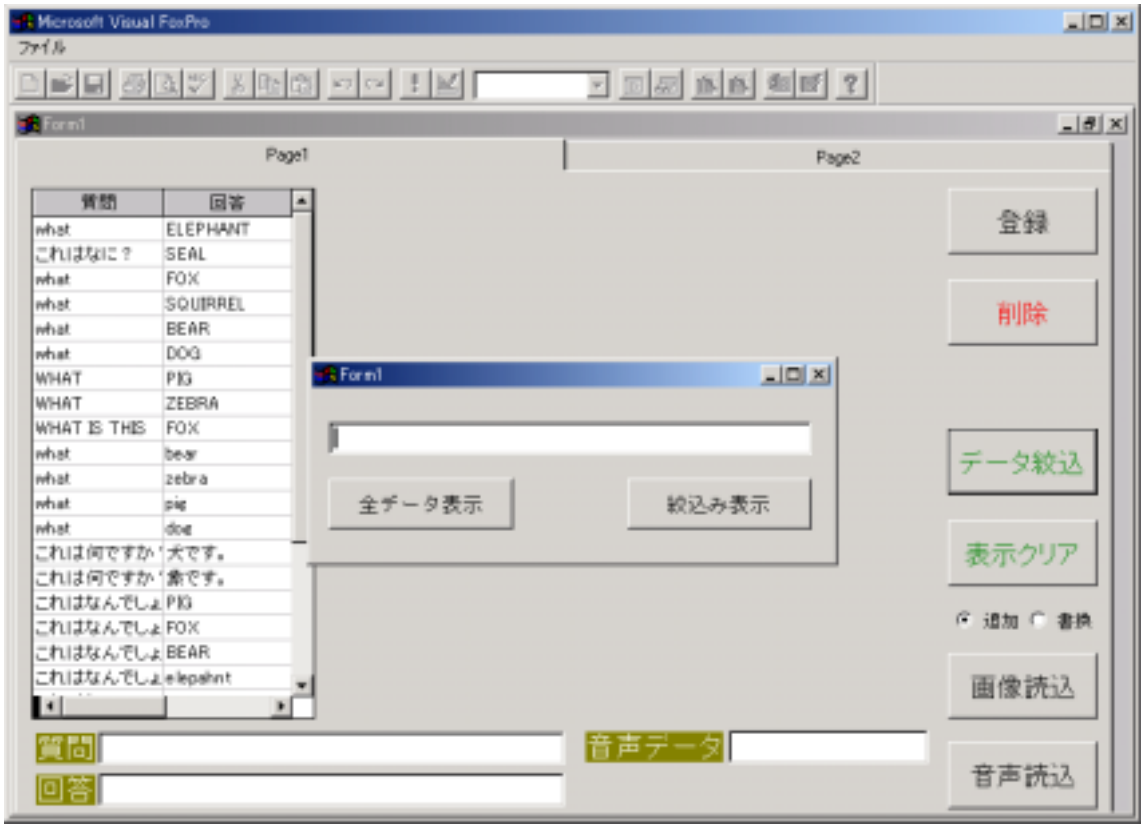
プログラム１



画面２メニューの作成

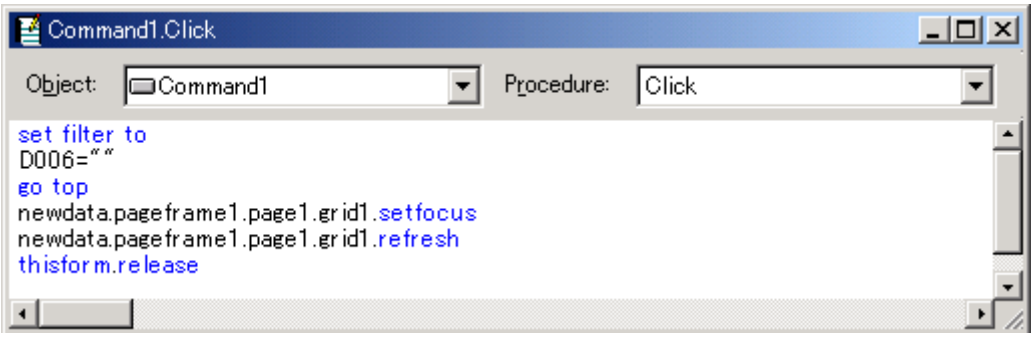


画面３メニュー実行ファイル（拡張子 mpr）の生成



画面４完成したプログラムを動かし、子フォームを表示した様子。

フォームのモードによって子フォームと親フォームが同時に操作できてしまう事がある。これはフォームのプロパティで WindowType を 1-Modal にすることで、子フォームが表示されているときには親フォームは操作できなくなる。プログラム１で親フォームを起動することで、子フォームから親フォームの制御も簡単に可能となる。フォーム上のボタンとかグリッドはそれぞれのフォームに所属するので、その制御はそれぞれのフォームのオブジェクト名に続くそれぞれのオブジェクト名で指定して行う。具体的にはプログラム２のような記述となる。

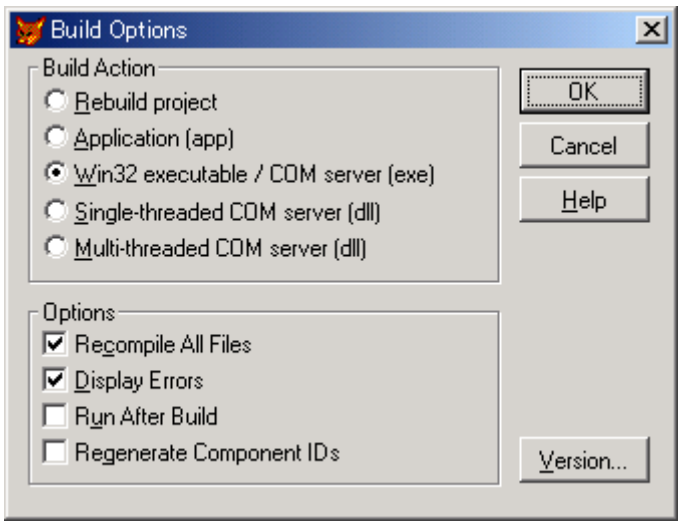


プログラム２フォームオブジェクトの制御

newdata.pageframe1.page1.grid1.setfocus　　&&　　グリッドにフォーカス

newdata.pageframe1.page1.grid1.refresh　　&&　　グリッドの表示再現

プログラム２の４～５行目では画面４の左側に表示されているグリッドの表示内容を変更する記述である。この親フォームにはページフレームがあり、その中の１ページ目を変更している。また thisform.release という記述も重要である。thisform は約束語であり、操作するオブジェクトのあるフォームを含む全てのオブジェクトは thisform で始まる記述で指定される。例えばプログラム２の４行目が操作しているフォームにあるオブジェクトの操作の場合は、thisform.glid1.setfocus である。ここでちょっと整理すると、VisualFoxPro でプログラムを作成する場合、データベースを含むデータの操作は Xbase 言語であり、これは MS-DOS 版の Xbase 言語や Arago for Windows と基本的に同じであり、フォームとフォーム上のオブジェクトを操作する場合の言語仕様は VisualFoxPro 独特のものになる。Xbase 言語の仕様を理解していて、フォームオブジェクトの操作方法が理解されていれば、簡単なプログラムは作る事が可能と思っても間違いない。ここまでのプログラムを EXE 形式の実効ファイルにするには、画面１で Build を押して、Build Option を呼び出し画面５



画面５プログラムの Build

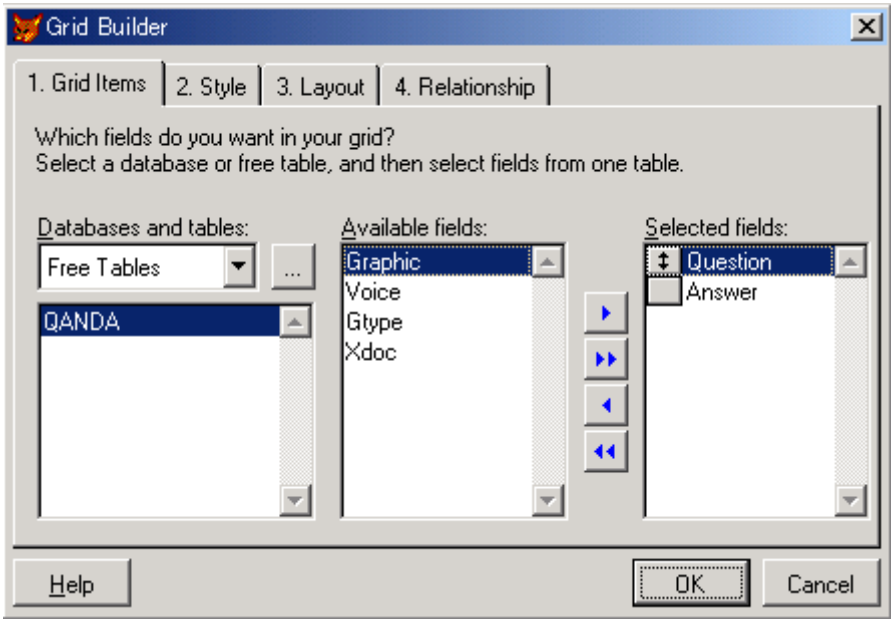
の設定で OK を押せば EXE 形式の実効ファイルは完成する。

「プログラムに様々な機能を追加する」

EXE ファイルが出来て、まあ一通りのプログラムは動かせても現在の水準から考えると機能性が十分なプログラムは構築できない。ここからは VisualFoxPro のテクニックの数々を紙面が許す限りご紹介しよう。

１．グリッドにデータベースの内容を表示する。

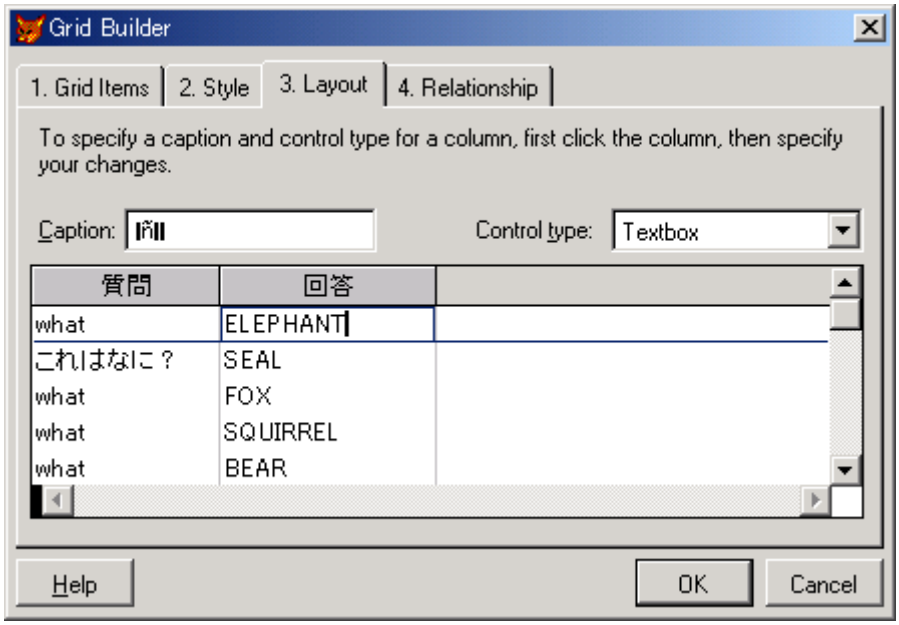
画面４の左端のグリッドにはデータベースの内容がそのまま表示されている。これはグリッドではなくてリストボックスでも良いのだが、複数のデータフィールドを表示するにはグリッドが最適である。グリッドにデータベースの内容を表示させるには、最初にグリッドをフォーム上に置いて、そのグリッドをドラッグしてマウスを右クリック→Builder ボタンを押す。画面６が現れるので、データベースの選択、表示するフィールドの選択を行い、



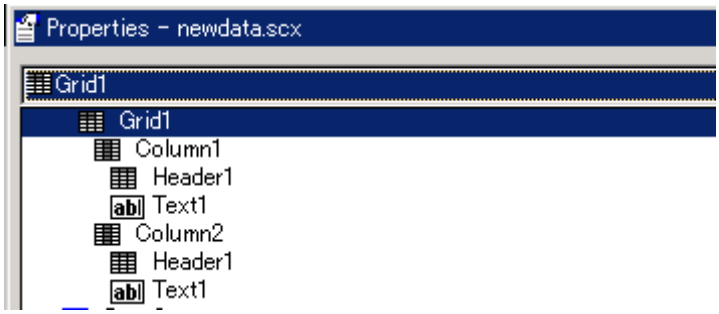
画面６グリッドの設定

次の Style は抜かし Layout タブを見る。ここでは表示するフィールド毎にそのフィールドのタイトルを決定する。ここでは日本語表示となっているが、最初にグリッドをドラッグしてマウスの右ボタンを押した時に表示されるプルダウンから Properties でフォントを日本語に設定する必要がある。特にグリッドについては Properties を設定する場所が複数存在し、何処に何を設定するかがポイントになる。画面８ではグリッドの設定の種類を見ることが出来る。ここでの設定の幾つかは画面６グリッドの設定で終了する。だが例えばグリッドに表示されるデータベースのデータをマウスのクリックで選択するような処理では、グリッドの Text1 の設定を行う。Properties をデフォルトから変更すると項目がボールドに表示が変わる。特にコマンドを記述するような項目は[User Procedure]と表示される。通常は[Default]である。画面９では既に修正済みの Text1 をダブルクリックした場合の Procedure を示している。

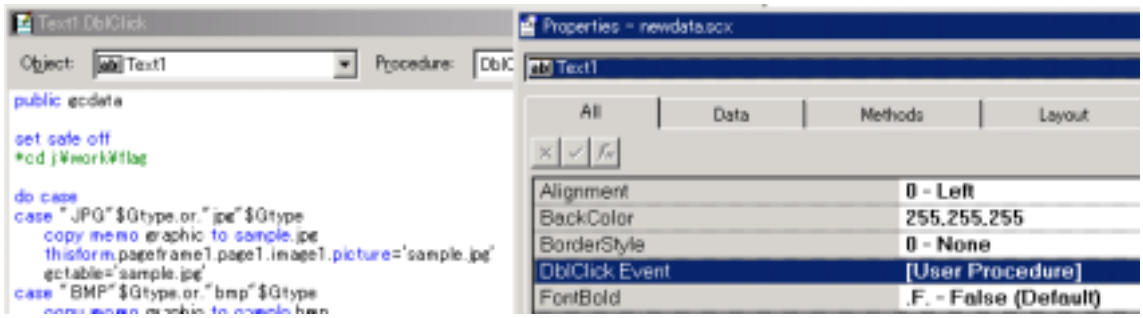
２．オリジナルのクラスを作成する。



画面７グリッドの Layout を変更する。

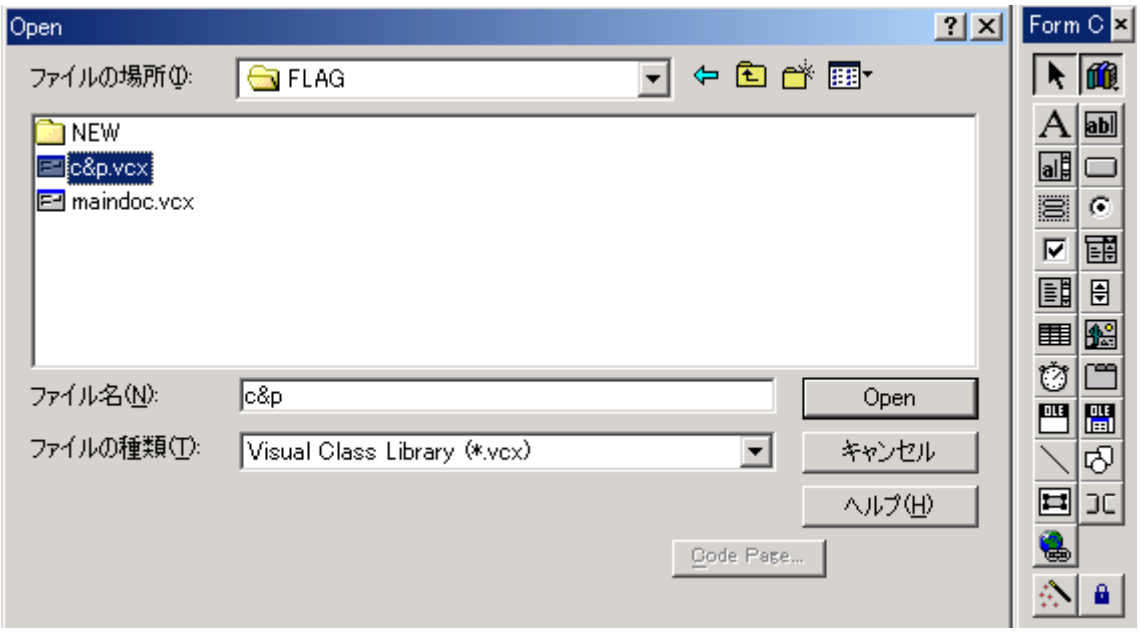


画面８グリッドの Properties の種類

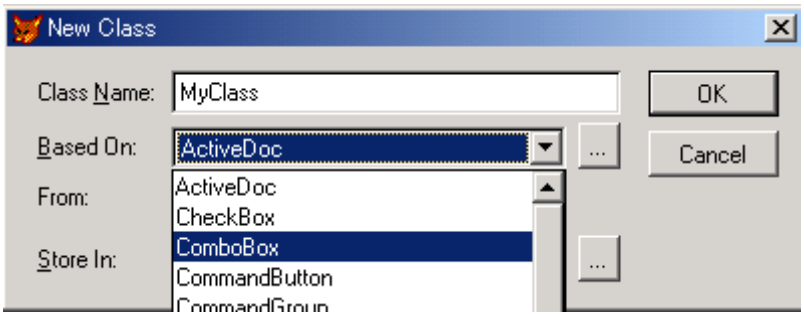


画面９グリッド Text1 の DblCkick を設定

このテクニックは VisualFoxPro としては中級ぐらいのもので、決して難しいものではない。だが覚えると癖になるというか、便利で色々作りたくなるテクニックである。最初に画面 10 をご覧頂きたい。画面 10 ではフォームのオブジェクトツールボックスで右上のライブラリーボタンを押して、このツールボックスにはないツールを呼び出そうとしている様子を示している。ここで自分で作成したオリジナルのクラスを呼べるのだが、これをどうやって用意するのか解説しよう。



画面 10 クラスライブラリーの呼び出し



画面 11 クラスを新規に作る

クラスを作るにはプログラムマネージャーの Class Libraries を選んで New ボタンを押す。すると画面 11 のように New Class が表示されるので、ここで新たに作る Class Name を入力して、Based On で自分が新たに作りたい機能を含むオブジェクトを選択する。ここまで来ると勘の良い方は、既にあるツールボックスのツールに機能を追加するのだと理解出来ると思う。例えば VisualFoxPro 自体のテキストボックスやエディットボックスにはマウスの右クリックで文字列をカット＆ペーストするような機能が標準ではない。この機能はテキストボックスやエディットボックスに個別に設定する事も可能なのだが、それでは全てに同じ事を都度しなくてはならない。ツールバーに最初からその機能を持ったテキストボックスやエディットボックスが存在すればそれを選んでフォームに貼り付ければ済んでしまう。文字のフォントやサイズも最初に標準化してしまえば、一々変更する必要はない。また一度フォームに貼り付けたオブジェクトでも、後でクラスの内容を書き換えれば、全ての設定を一気に変更可能だ。フォーム上のテキストの色をまとめて変更したいとか、サイズやフォントの変更も実に簡単だ。画面 12 では実際に作ったクラスの内容を示している。User Procedure の内容は DO edtshort.mpr WITH THIS としている。このメニューは

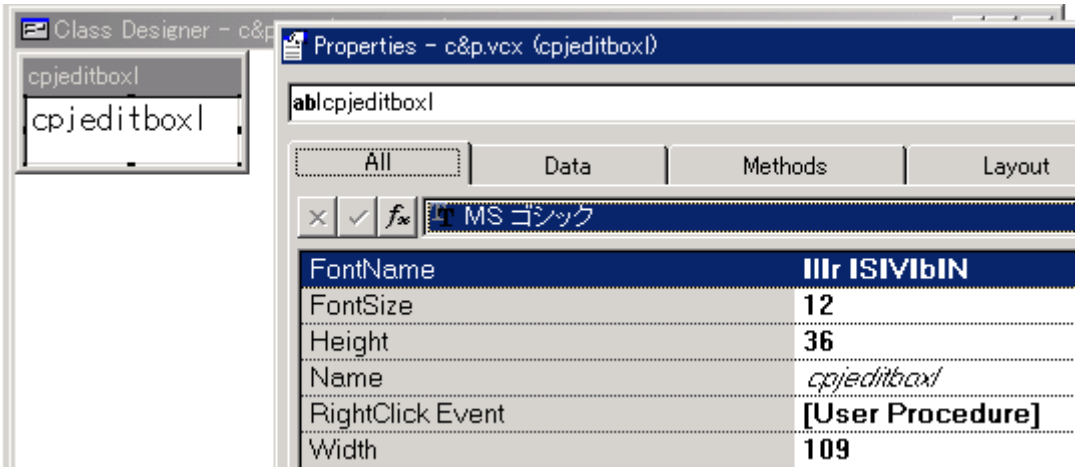
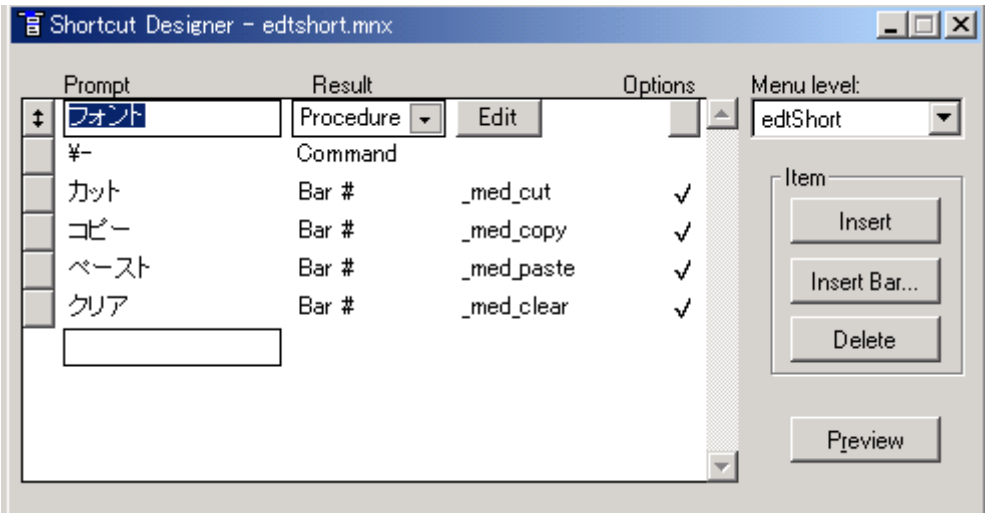


図 12 完成したオリジナルの editbox、フォントは日本語でサイズを変えた。RightClick Event（マウスの右クリック）でポップアップのメニューが出るようにした。

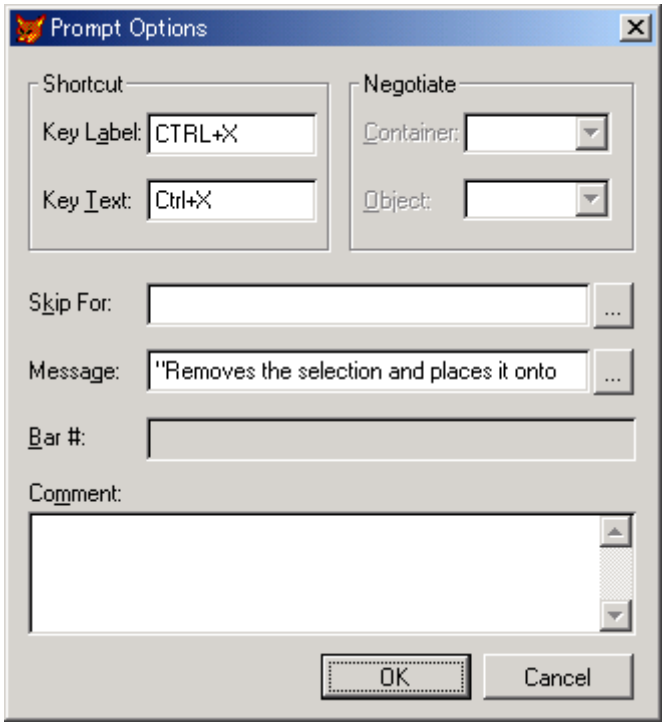


画面 13 カットアンドペーストを可能にするポップアップメニュー

カットアンドペーストのサンプルメニューが VisualFoxPro の CD にサンプルで付属している。サンプルを参照するのも VisualFoxPro 理解への近道である。カットやコピーの機能はそれぞれのメニュー項目の Option に記述されている。例えばカットは画面 13 のように記述されている。これはコピーやペーストも同様である。自分で作ったクラスはそのままプロジェクトマネージャーに登録されるが、登録されなくても後から呼び出すことが可能だ。

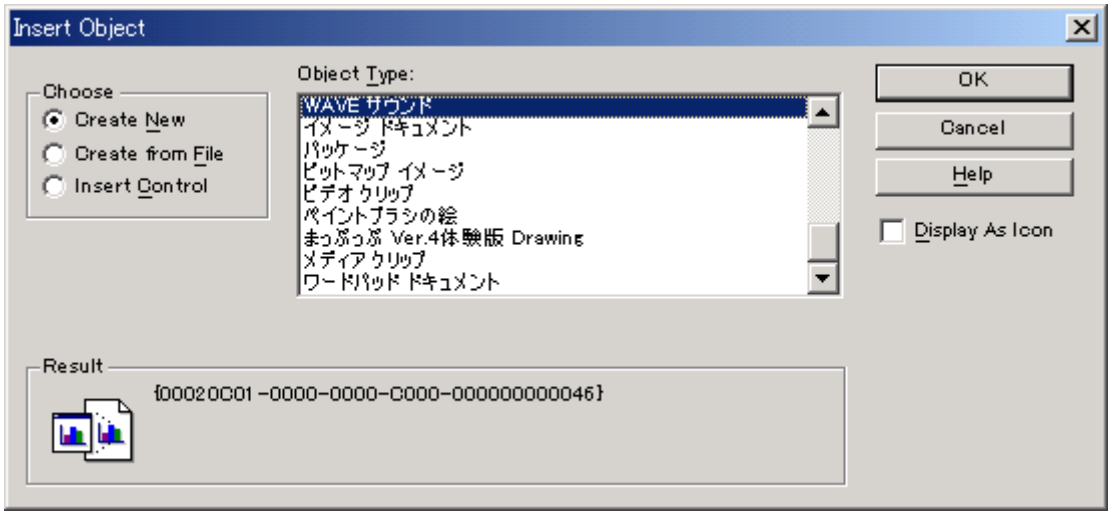
３．プログラムで音を出すには

VisualFoxPro で音を出す方法は幾つかあるが、簡単なのは Xbase 言語の拡張機能を利用する方法である。過去の Xbase 言語では SET BELL ON|OFF だけだったのに対して VisualFoxPro では SET BELL TO ????.WAV として WAV ファイルの音声を再生できる。そして実際の再生には?? CHR(7)を利用する。これだけで指定した WAV ファイルの再生が可能になる。もう一つは Olecontrol を利用する方法だ。この方法を見つけた理由は WEB アプリケーションとして VisualFoxPro を利用した場合に SET BELL TO ????.WAV では



画面 13 カットの機能を記述。

インターネットのホームページに VisualFoxPro を置いて音を出すことが出来なかったのである。そこで試行錯誤の末、Olecontrol を VisualFoxPro に入れることで、音の再生が可能となった。この方法は意外と面倒で、確かに Olecontrol を操作すると音は出るのだが、それをプログラムの中から操作する方法を見つけるのに半日ぐらい掛かった。でも動いてしまえばどうという事はない。Olecontrol をフォームに置くためには最初にフォームコントロールボックスの中の Olecontrol を押して、フォーム上に範囲を決める。そうすると画面 14 のパネルが表示されるので「WAVE サウンド」を選択する。



画面 14 フォームへの Olecontrol の挿入

そしてこの Olecontrol にサウンドファイルを設定する。Olecontrol をマウスでドラッグし

右クリックでサウンドレコーダードキュメントの編集を選ぶと図 15 のレコーダーが表示されるので、メニューの編集で設定する WAV ファイルを選ぶ。

図 15 サウンドオブジェクトの設定

このオブジェクトでプログラム上から音を出すことは次の 1 行で可能となる。

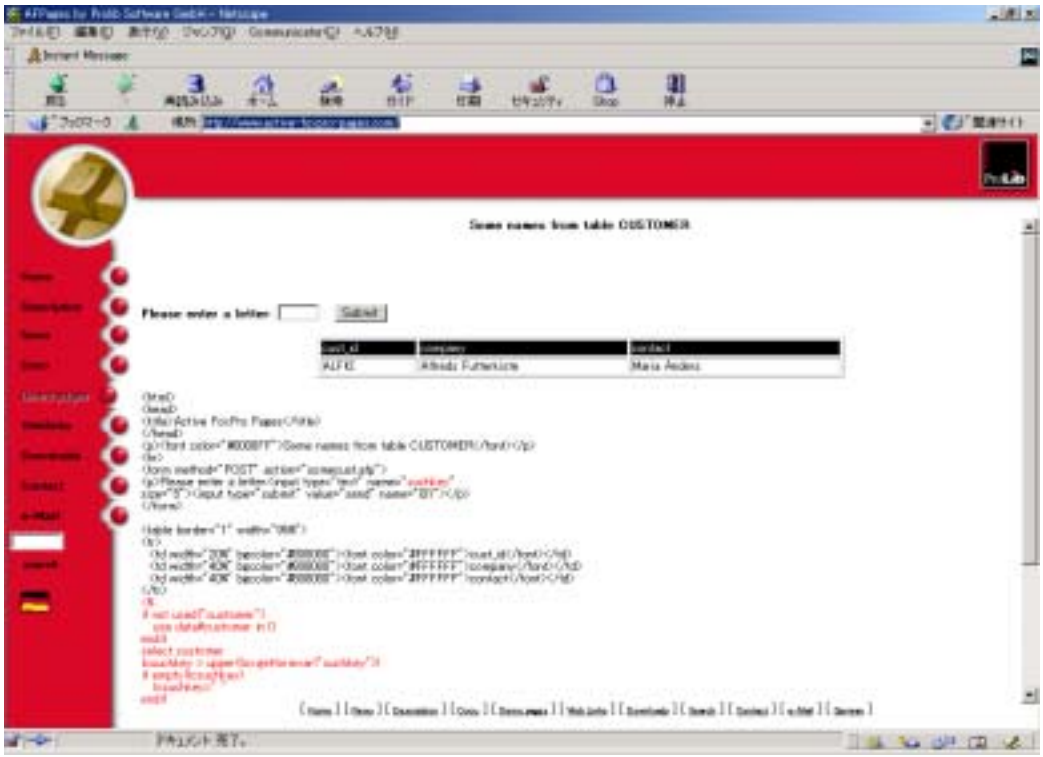
thisform.pageframe1.page2.olecontrol1.doverb (0)

早い話、olecontrol1 の設定を doverb(0)とすれば良いのである。doverb の設定は doverb(0)で実行、doverb(-1)で図 15 のオブジェクトが表示される。以上によって WAV ファイルは再生されるが、EXE のアプリケーションなら SET BELL TO ????.WAV の方がレスポンスは良く、アプリケーションも軽くなる。ただ WEB アプリケーションとして VisualFoxPro を使う場合にはどうしても Olecontrol が必要になる。

「完成度の高いアプリケーションを作成する」

安定していて完成度の高いアプリケーションを作成するにはどうしたら良いか？全てのプログラマーが悩む問題である。VisualBasic 等は手軽にアプリケーションを作れてしまうが、安定度やデバッグ、変更といった後処理にはウィークポイントがあると多くの技術者が指摘する。ACCESS は小規模なシステムには使えても全社規模となると使いにくく、Java は WEB には良くても、データベース処理が大規模になったようなシステムの例がなく、処理の少ないフロントエンド向きに思える。Delphi には死角はなさそうだが、ローカルエリアで WEB アプリケーションを構築した場合は CGI を使う Delphi よりも VisualFoxPro の方が遥かに処理は早い。VisualFoxPro に残された問題はインターネットの WEB サーバサイドだけでデータベースを更新し、データ処理する問題である。これについてはまだ十分な結論が出ていないのだが、実際にこれを実現している WEB ページが海外にはある。画面 16 はドイツのサイトで AFP（http://www.active-foxpro-pages.com/default.htm）そして、最も技術力が高いとされる画面 17 の WEST-WIND（http://www.west-wind.com/）である。

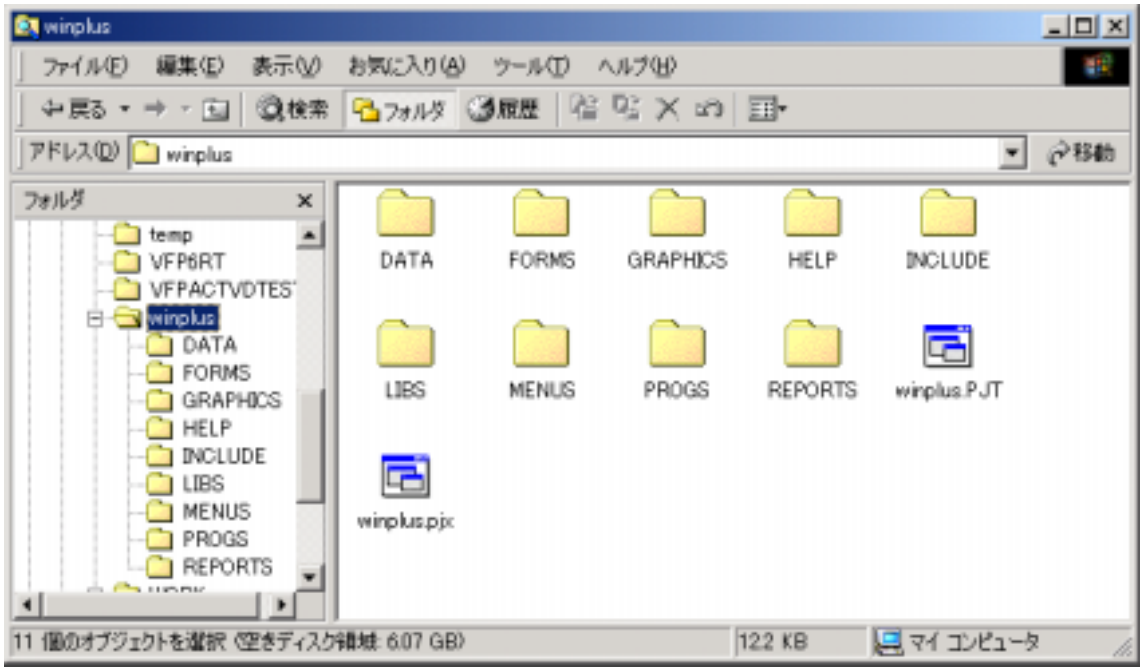
話を元に戻そう。現在のコンピュータ言語は一昔前のように単一の環境に対応するのでは許されなくなり、様々な環境に対応できなくては認められなくなってきており、プログラムの作りやすさや安定度にはマイナスの要因になっているのだが、VisualFoxPro には一つの解決策として安定したアプリケーションのベースをある所まで自動的に生成する Application Wizard が用意されており、VisualFoxPro5.0 を継承するバージョンと 6.0 独自バージョンの二つが用意されている。5.0 互換フレームワークの方が分かりやすくすっきりしている。Application Wizard5.0 を実行し数十秒が経過するとプロジェクトが完成する。



画面 16VisualFoxPro で WEB サーバデータベース処理を実現する AFP

画面 17 こちらも VisualFoxPro の WEB 化に力を入れている。

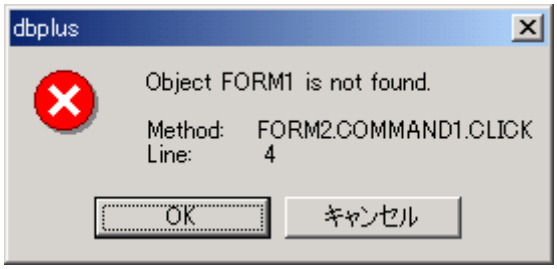
Application Wizard の実行時に必用な手続きは適当なプロジェクトの名所を決めるだけだ。



画面 18 作成されたフレームワークとディレクトリ

Application Wizard で作られたアプリケーションフレームワークでは関連ファイルが画面 18 のようにフォルダで整理され、多くの機能が完成した形で用意される。特にエラー時の対処方法が明確になり、プログラムを構築している途中では非常に役に立つ。また実行時のプログラムではエラーでハングしなくなる。あるボタンを触ると落ちてしまうが、他には問題ないというような場合にはこれは有り難い。そしてその手法もこのフレームワークから学ぶ事が可能だ。次のコメントを含む４行はフレームワークを作成後に自分のアプリケーションを登録する部分である。ここでは最初に起動するフォームの名称、使うメニュー、フォーム名を記入しておく。

```
*-- Configure application object.
goApp.SetCaption("dbplus")
goApp.cStartupMenu="menu1"
goApp.cStartupForm="newdata"
```



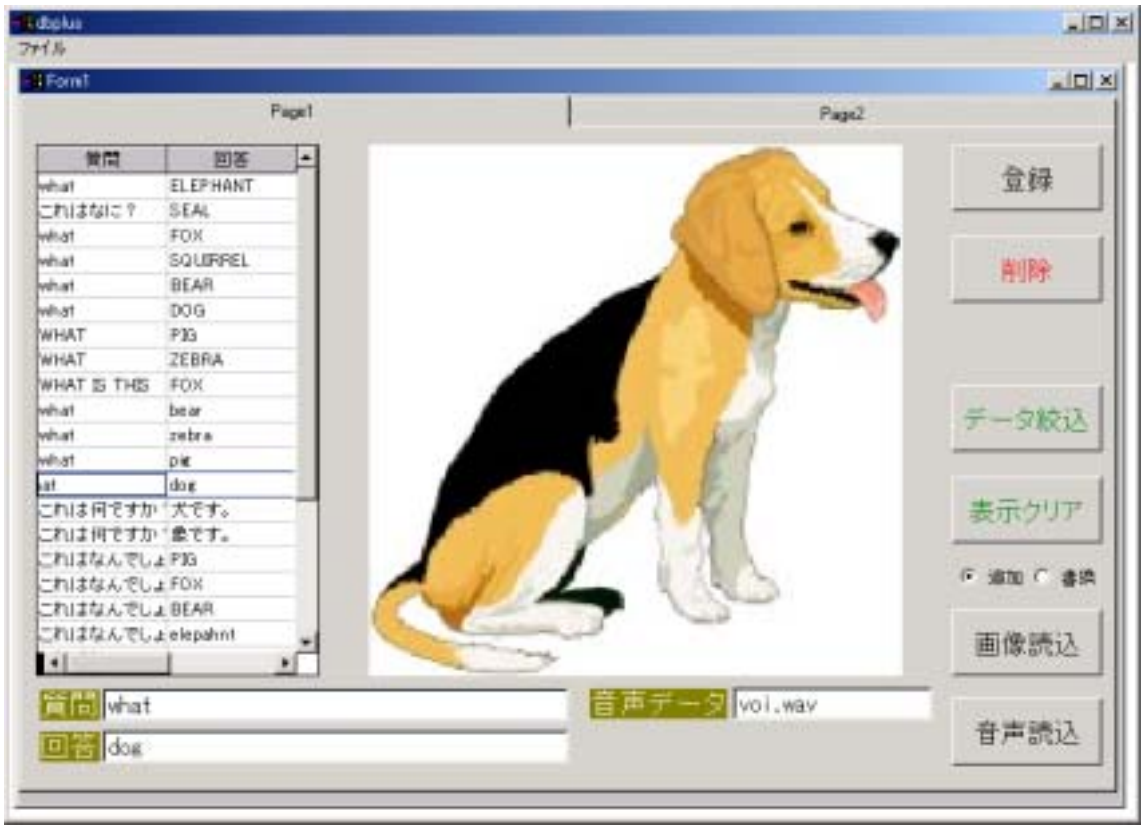
画面 19 実行時エラーの表示

実行時にエラーを起こした場合の位置と内容を図 19 に示す。ここで「OK」を押すとエラーを避けて、プログラム自体は停止しない。最終的にはこうしたエラーは出なくなる事が

理想だが、プログラムのバグ以外でもデータファイルが消失した場合などにシステムがハングする問題を回避できる。特に有り難いのはエラートラップを全く意識せずにシステム構築が可能になってしまう点だ。

「VisualFoxPro は軽い」

マイクロソフトのアナウンスによると VisualFoxPro7.0 は Visual Studio.NET の次期バージョンには含まれないらしいが、夏ごろまでにはリリースされるらしい。そして既にβ版を試した方の話によると、VisualFoxPro6.0 と 7.0 の違いは僅かだという。他の VisualBasic 等の汎用的な開発環境が高機能でかつ重くなっている事を考えると、VisualFoxPro の存在は特種だと言える。今時 486 の環境でも十分なパフォーマンスが得られ、.NET では共通の環境で全ての開発システムを統合するはずだった方針が、パフォーマンスを重視する VisualFoxPro だけは独自の DLL を継承するという話である。誌上で幾らご説明しても VisualFoxPro のスピードや軽さは全く理解していただけないので、VisualFoxPro で作成したアプリケーションを Vector に準備している。内容は画像と音声ファイルを管理するデータベース管理ソフトである。（画面 20）



画面 20 画像と音声ファイルを管理するデータベース DBPLUS（仮称）

このソフトは画像と音声のデータを VisualFoxPro のデータベースに入れて管理するソフトである。画像や音声のデータを管理する場合、画像や音声のデータをフォルダ内で複数管理する事が多いが、このソフトでは VisualFoxPro のデータベース数個でほぼ無制限に画

像を管理できる。データを他のアプリケーションで利用する場合は、検索して表示すると特定の名前でフォルダに書き出されるので、これをそのアプリケーションから利用できる。ただデータが管理出来るだけでは面白くないので、このデータベースを利用した学習機能なども付属したいと考えていて、その部分が完成すれば Vector に登録する予定であるが、筆者のサイトには３月中に登録されると思う。VisualFoxPro の実力を少しでも理解して頂ければ幸いだ。

「何処へ行くのだろうか？」

パソコンを初めて操作したのは 20 年以上前の話になる。それから数年して自信がユーザーになり、ソフトウェアの開発を始め、Xbase 言語と出会って 15 年が経過する。Xbase 言語はバッチ処理だとその時代に席を並べていた友人に悪口を言われ、C 言語でちょっとした開発をしたりしていた時期もあるが、サザンの Xbase 言語である QuickSilver と出会ってからはエンドユーザーシステムには Xbase 言語以外は考えられなくなった時期もあった。Windows に環境が移行してから多くの仲間が VisualBasic に移行したが、そうした彼らの感想はユーザーさえ認めれば Xbase 言語で作りたいというものである。ただオブジェクト指向から程遠い Arago for Windows を VisualBasic のように使うのは困難だ。VisualFoxPro はオブジェクト指向の Arago for Windows と呼べるもので、VisualBasic や Delphi 等に慣れていれば取っ付き易い言語である。VisualFoxPro から J++や Excel を操作する方法や HTML を出力する手法も分かって来たので何処かでまた書きたいと考えている。VisualFoxPro はそれ程に守備範囲が広く、軽く早いシステムなのである。ただ場合によってはやはり Arago for Windows の方が生産性は高く、デバッグも楽だと思うことがある。はっきり言えばどちらでも良いのではないだろうか？Arago for Windows のデータベースは Xbase 言語だけだが、VisualFoxPro なら Oracle や SQL Server にもアクセス出きるのだから、VisualBasic で行き詰まったら VisualFoxPro を使うとか、それで表現力不足なら J++を使えば良いのである。勿論、Arago for Windows と VisualFoxPro を混在しても良い。少なくともレポートやバッチ処理の生産性は VisualFoxPro が群を抜いている。その生産性とスピードを知ったら諦めていた全てに可能性が見えてくるだろう。あのマイクロソフトが VisualFoxPro だけは.NET の共通モジュールを使わないという。これは VisualFoxPro のスピードを重視しての処置だ。何が重要かをはっきり認識している開発システムはあれだけの企業としては珍しい。米国では哲学ある言語として VisualFoxPro は評価されている。何時の頃からかパソコンの世界はマニアやパワーユーザーのものから、バブル期の不動産のようになってしまった。これは大変に不幸な事である。パソコンには現場で苦労するパートさん達を救う所に本来の目的がある。これを思い出し実践することは日本の将来を救う鍵になると信じている。本誌では２年間に渡ってお付き合い頂いた事に厚くお礼を申し上げると共に、また何かの形でお会い出来ると思うので更なる応援をお願いしたい次第である。