Clarion Magazine

Converting The Inventory Example - Calling Stored Procedures (Part Three)

In the conclusion to this three part series, Ayo Ogundahunsi demonstrates how to use the supplied template to call stored procedures.

Posted Thursday, November 07, 2002

A Class For The ASCIIng

ASCII files are commonplace in programming, but their use in Clarion is generally more complicated than in other languages because Clarion can only do ASCII file I/O via an ASCII file driver declared file. Declaring such a file can become quite a nuisance, and breaks Konrad Byers' golden rule of never having to program the same thing twice. Enter the any ASCII file class.

Posted Friday, November 08, 2002

Data Structures and Algorithms Part XI - Binary Search Tree Indexing

Have you ever wanted to key index a text file so that you can fetch a single record directly? In this article Alison Neal shows how you can do this using a Binary Search Tree.

Posted Friday, November 08, 2002

Quick COM Using A Windows Script Component

Everyone knows that Clarion can do COM, but did you know that you can interface a Clarion program *to*COM? Wayne Price shows how to use a Windows Script Component to create a Clarion program that's usable by any programming language that can call COM objects.

Posted Wednesday, November 13, 2002

Determining Gender With Clarion (Part 1)

Inspiration can come from the strangest sources. An automatic complaint letter generator sends Geoff Robinson on a quest to create a gender guessing function using regular expressions. Part 1 of 2.

Posted Thursday, November 14, 2002

SQL Data Types Comparison

News

- Business Rules Engine Beta Testers Required.
- QuickBooks SDK 2.0
- Hotfix For ImageEx2 Beta 2
- Helpmaker Website Change
- MySQL Browse Class Updated
- ImageEx2 Beta 2
- WebUpdate Template Corrected
- MySQL Help Document
- List & Label v9 Demo
- IconsXP Specialty Groups
- Date Handling Class
- CapeSoft Updates
- File Manager 3 Beta Progressing
- Office Inside
- CapeSoft's Replicate Beta
- MySQL Page Loaded ABC Browses

So you think one SQL database is pretty much like another? Certainly not when it comes to available data types. David Harms explores the minor and major data type differences between Firebird, MySQL, PostgreSQL, and Microsoft SQL Server.

Posted Friday, November 15, 2002

Determining Gender With Clarion (Part 2)

Inspiration can come from the strangest sources. An automatic complaint letter generator sends Geoff Robinson on a quest to create a gender guessing function using regular expressions. Part 2 of 2.

Posted Tuesday, November 19, 2002

Staying Connected On The Road

If you're like a lot of Clarion software developers, you spend at least some time on the road, and, you no doubt have your favorite way of staying connected to the online world. If you stay in hotels, or with friends or family, a phone line is never far away. But what if you travel by recreational vehicle (RV)? Ned Reiter explores the options, from dialup to satellite.

Posted Thursday, November 21, 2002

Books Reviewed: Getting Your ASP In Gear

No, there's no ASP book by that title, at least not yet. Instead, David Harms reviews two books from O'Reilly on the subject of ASP development.

Posted Friday, November 22, 2002

Introducing PostgreSQL (Part 1)

PostgreSQL, a long-time open source rival to the MySQL database, isn't particularly well known among Clarion developers. In part, that's because until recently the only way to run PostgreSQL on a Windows machine was by using the Cygwin Linux emulation. A beta of a native Windows version is now available, and that's prompted David Harms to begin this new series on PostgreSQL.

Posted Thursday, November 28, 2002

Securing Remote Database Connections With SSH Tunneling

If you've ever had to connect to a database across the internet, you've probably worried about the security of your data. Even if your database and driver don't support encryption, you can easily secure your connection by tunneling it through the SSH protocol. David Harms shows how it's done.

Posted Thursday, November 28, 2002

Clarion SOAP Revisited

The Simple Object Access Protocol (SOAP) has been covered before in Clarion Magazine, but only from the Clarion end. In this article, Radventure's Erik Pepping shows how to create the server in C# and the client in Clarion.

Posted Thursday, November 28, 2002

Looking for more? Check out the **site index**, or **search the back issues**. This site now contains more than 700 articles and a total of over a million words of Clarion-related information.

Available

SCA Micro Templates

ThinkData Releases qbFUSE QuickBooks COM Automation

Image-XChange SDK Released

Taboga Software Holiday Schedule

ImageEx 2 Preview Demo

Search the news archive

Converting The Inventory Example - Calling Stored Procedures (Part Three)

by Ayo Ogundahunsi

Published 2002-11-07

In my last article, Converting The Inventory Example - Calling Stored Procedures (Part Two), I showed you how to integrate your application with SQL Server using the supplied template. Now that I have a working application based on the template, I want to demonstrate how to call stored procedures.

Before I look at calling stored procedures, it is important that I mention some tools that will help you troubleshoot activity between your Client application and SQL Server. For someone new to SQL, it can be a daunting task when your application fails and you have no idea about what is going on. Two useful programs that can help you to debug SQL activity are Clarion's TRACE.EXE and the MSSQL Profiler.

TRACE.EXE

TRACE.EXE is a Clarion program that allows you to trace input/output from any Clarion file driver. It is located in your Clarion installation's `bin` subdirectory. The shortcut to this program is usually added to your Clarion menu under the Tools submenu. When you run TRACE.EXE you'll see the window shown in Figure 1.

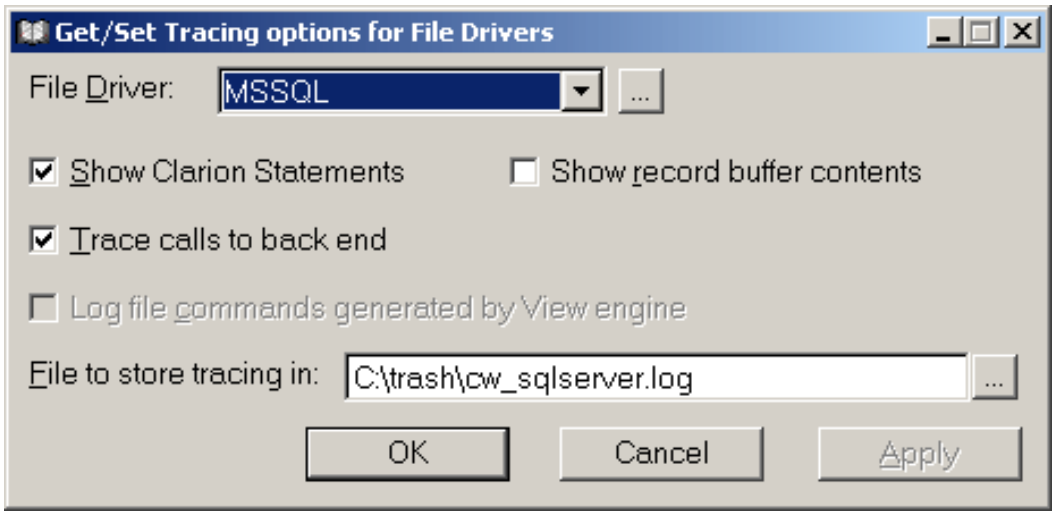


Figure 1. Setting Trace Properties

Remember to stop the trace once you are done using it, because this file can really grow big and will slow down your application considerably.

The trace is in text form and can look cryptic sometimes, but with a little effort you shouldn't find it too difficult to read.

MSSQL Profiler

Profiler is a utility in SQL Server. Using this program you can log the actual SQL commands that are executed by your client application. If you use the Enterprise manager a lot, you might want to capture SQL code. Also, any SQL code executed after the profiler starts (that could be as a result of the `PROP:SQL` command, or any stored procedure) is captured by the profiler. for a point and click operation you just performed. The profiler is very useful in this regard.

You can run the profiler from the Enterprise manager, as shown in Figure 2.

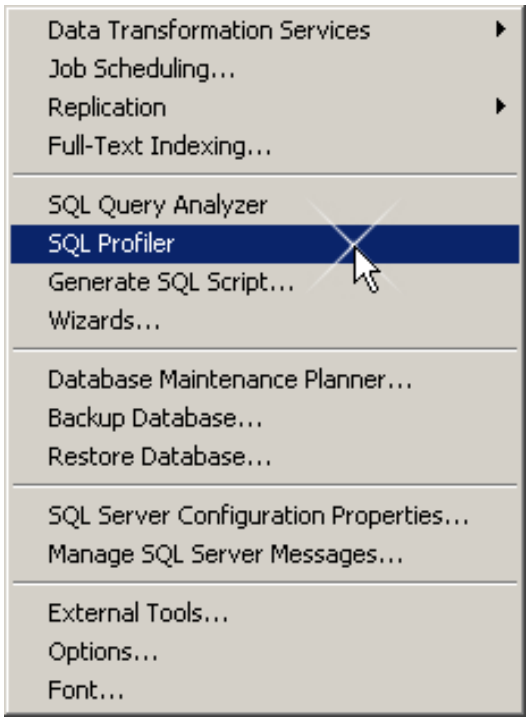


Figure 2. Calling Profiler from Enterprise Manager

Or, from the SQL Server menu, as in Figure 3.

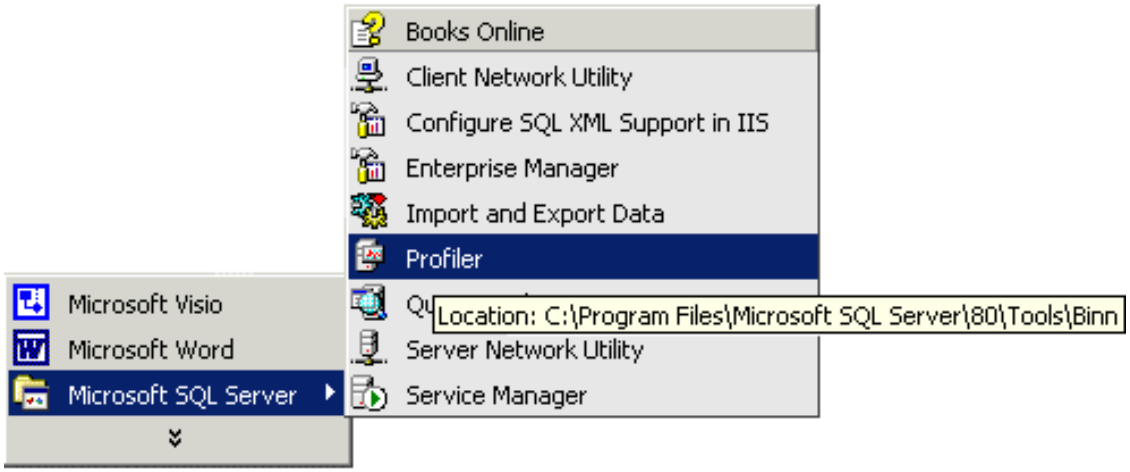


Figure 3. Calling Profiler from Program Menu

When you execute the stored procedure example in this article, the Profiler will capture an output as shown in Figure 4.

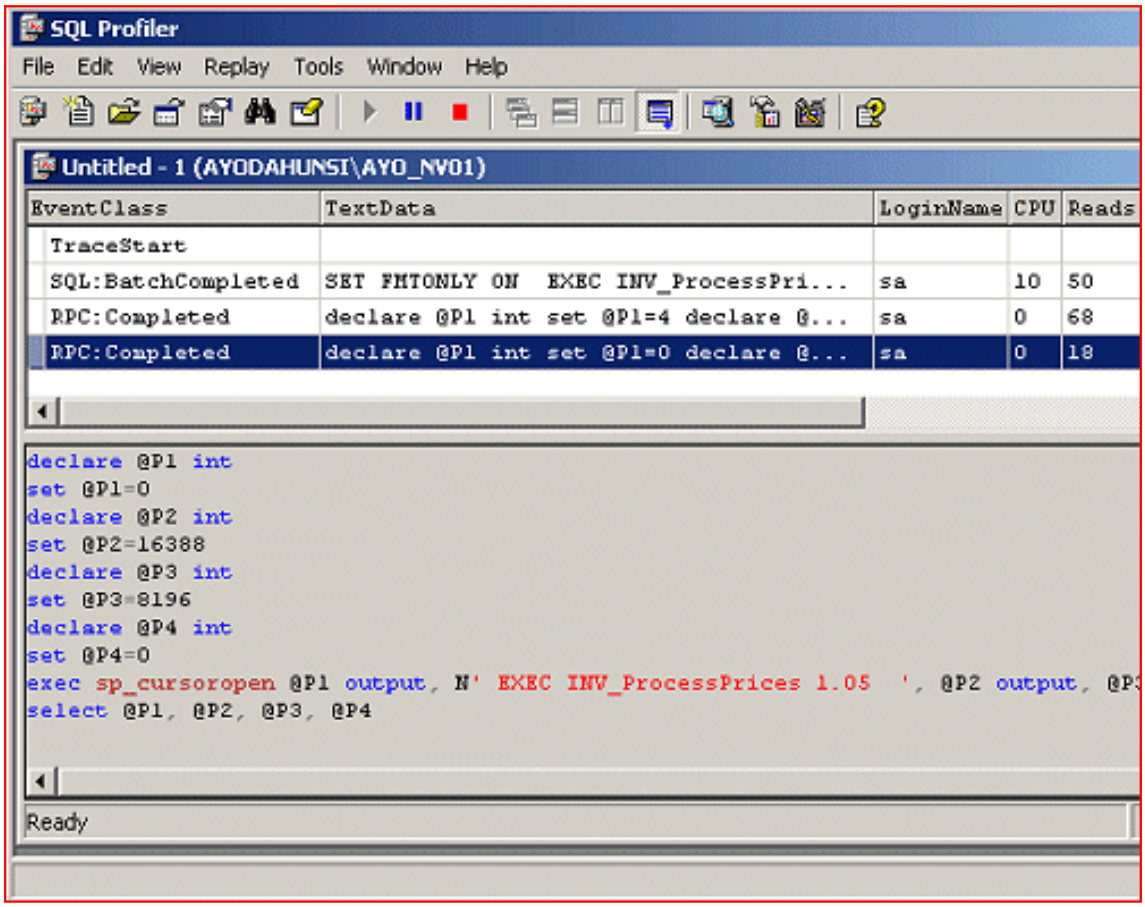


Figure 4. Viewing Trace from Profiler

The profiler is not available with Microsoft Desktop Engine (MSDE), a trimmed down version of SQL Server.

Why stored procedures?

One of the strengths of Clarion lies in the fact that you can roll out applications easily and quickly. From standard report/browse/form handling to automatic data type conversion to threading (this will probably change with the upcoming version 5.6), a lot is done behind the scenes. As a result, there is always the tendency to want to write as much as you can in Clarion. Unfortunately, this might not be the best approach when you are using Clarion with a Relational Database System (RDBMS).

The first step in migrating an application to SQL Server is to get the application to run. After conversion, you need to examine parts of your code that result in frequent trips to the server, and then try to optimize by moving some processing to the server.

A stored procedure in simple terms is your code written in SQL syntax and stored at the server, where they become part of the database just as tables are part of the database.

There are a many books on SQL that explain more about stored procedures so I won't go into a

lot of detail here.

The ProcessPrices procedure

In `INV_SQL.APP`, there is the `ProcessPrices` procedure that is used to do a batch update on all product prices based on a specified percentage. If, after compiling the application as demonstrated in my last article, you try to run this procedure you get a Record Not Available error as shown in Figure 5.

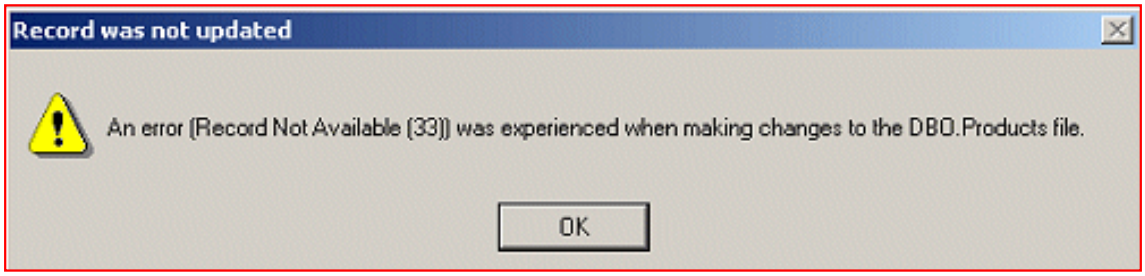


Figure 5. Error after conversion

One lesson to learn here is that straight conversion will not always guarantee your application will work out of the box. A lot of tasks handled by the template when you are using a flat file system might not work when you convert to SQL. I don't want to get into too many details about this particular error, but, if you uncheck **Use RI constraints on Action** and recompile, the error disappears and the update is done.

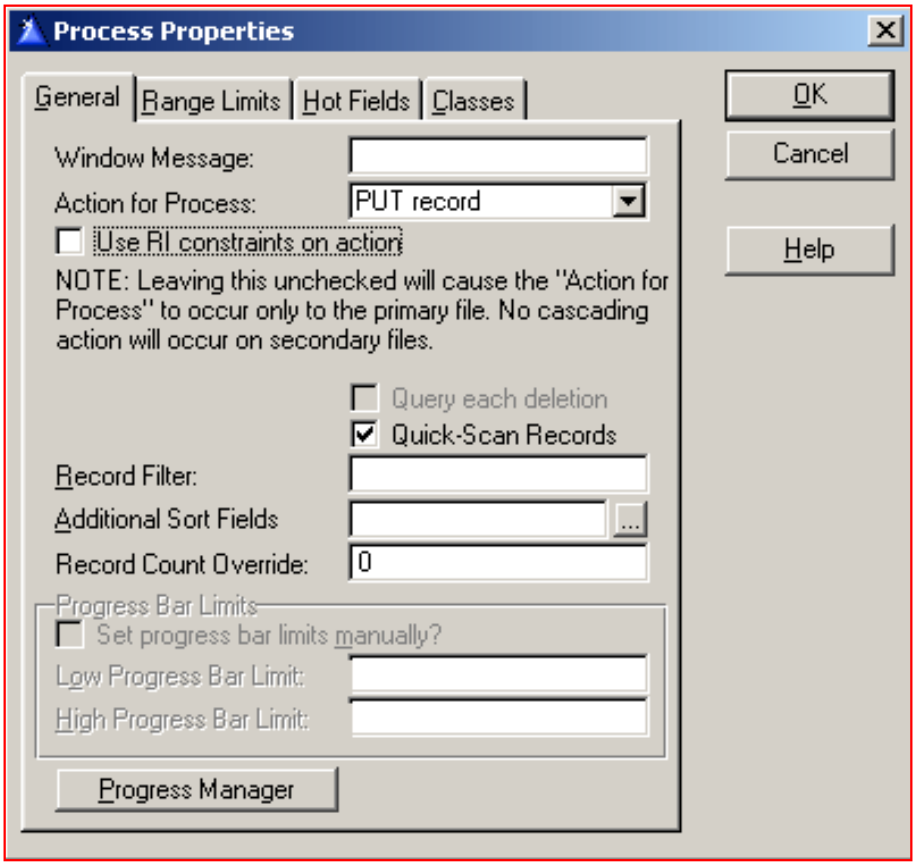


Figure 6. Process Template - Properties

Take a look at the cost in bandwidth of doing the update the traditional (flat file) way. If you look at the Clarion Trace program, you will see that the SQL UPDATE command was called repeatedly, once for each record in the PRODUCTS table.

```
NEXT(VIEW:416AA4:DBO.Products 2) Time Taken: 0.00 secs
POSITIONfile(VIEW:416AA4:DBO.Products 2) Time Taken: 0.00 secs
Closing Statement 1104f08 Time Taken:0.00 secs
Resetting Parameters Statement 1104f08 Time Taken:0.00 secs
Preparing Statement 1104f08 : UPDATE DBO.Products SET "PRICE" = ? ↵
  WHERE "ID_PRODUCTS" = ?  Time Taken:0.00 secs
Binding ? 1 for input with C type CHAR as 3 for Statement ↵
  1104f08 Time Taken:0.00 secs
Binding ? 2 for input with C type SLONG as 4 for Statement ↵
  1104f08 Time Taken:0.00 secs
Executing prepared Statement 1104f08 Time Taken:0.00 secs
PUT(VIEW:416AA4:DBO.Products 2) Time Taken: 0.01 secs
```

This code (extracted from the Trace file), is executed 32 times (there 32 records in the PRODUCTS table), which means the client machine was the one initiating and executing the update logic. Even though this might not seem quite a problem, imagine what happens when you try to update about 200,000 records. This means this command will be called 200,000 times!

This kind of repetitive client/server interaction negates the purpose of developing an application to run in a SQL environment. Assuming I want to increase all prices by 5%, the best thing would have been to send a SQL command like this:

```
UPDATE PRODUCTS
SET PRICE = (1.05 * PRICE)
```

However, if I embed this code in my Clarion application, and call it via PROP:SQL, no other application will be able to use it.

A better approach is to create a stored procedure that receives a parameter for the price factor (i.e. 1.05). In that way any other program, such as one written Visual Basic, or perhaps an ASP web application, can also call this update.

In short, one very big advantage of using stored procedures is that you can remove most of your business logic from your Clarion application and make it part of your database.

That’s the theory; now it’s time to replace the ProcessPrices procedure with a stored procedure.

The ProcessPrices stored procedure

The code to create the stored procedure to update the product prices is as follows:

```
CREATE  PROCEDURE dbo.INV_ProcessPrices
(
    @PercentageIncrease decimal(7, 2)
)
/*
Object:      INV_ProcessPrices
Description: Allows you to increase the prices of all
              items in inventory
Usage:       INV_ProcessPrices @PercentageIncrease=1.5
              -- 5% increase
Returns:     (None)
Author:      Ayodele Ogundahunsi  Email: ayodele@dolasoft.com
Revision:    1
Example:     INV_ProcessPrices @PercentageIncrease=1.5
Created:     2002-09-30.
*/
AS
SET NOCOUNT ON
BEGIN
    UPDATE PRODUCTS
    SET COST = (COST*@PercentageIncrease)
END
RETURN 0
```

Note that it is not a very good idea to name your stored procedure starting with `sp_`. While `sp_` is not a reserved word, it means "special" within the context of SQL Server. There are a lot of special system stored procedures shipped with SQL Server, and you don’t want to confuse them with your stored procedures.

Linking INV_ProcessPrices to Clarion

Now that I have created the stored procedure `INV_ProcessPrices`, I will need to link it with Clarion. I have included an extension template in the downloadable source that makes this easy. This extension template has been designed to create a procedure that wraps all calls and interfacing with SQL Server all contained in one single place. This approach makes it possible to create an external procedure in another EXE or DLL that calls a procedure created here. As a standard, I will call my procedure the name of the stored procedure I am calling, in this case, `INV_ProcessPrices`.

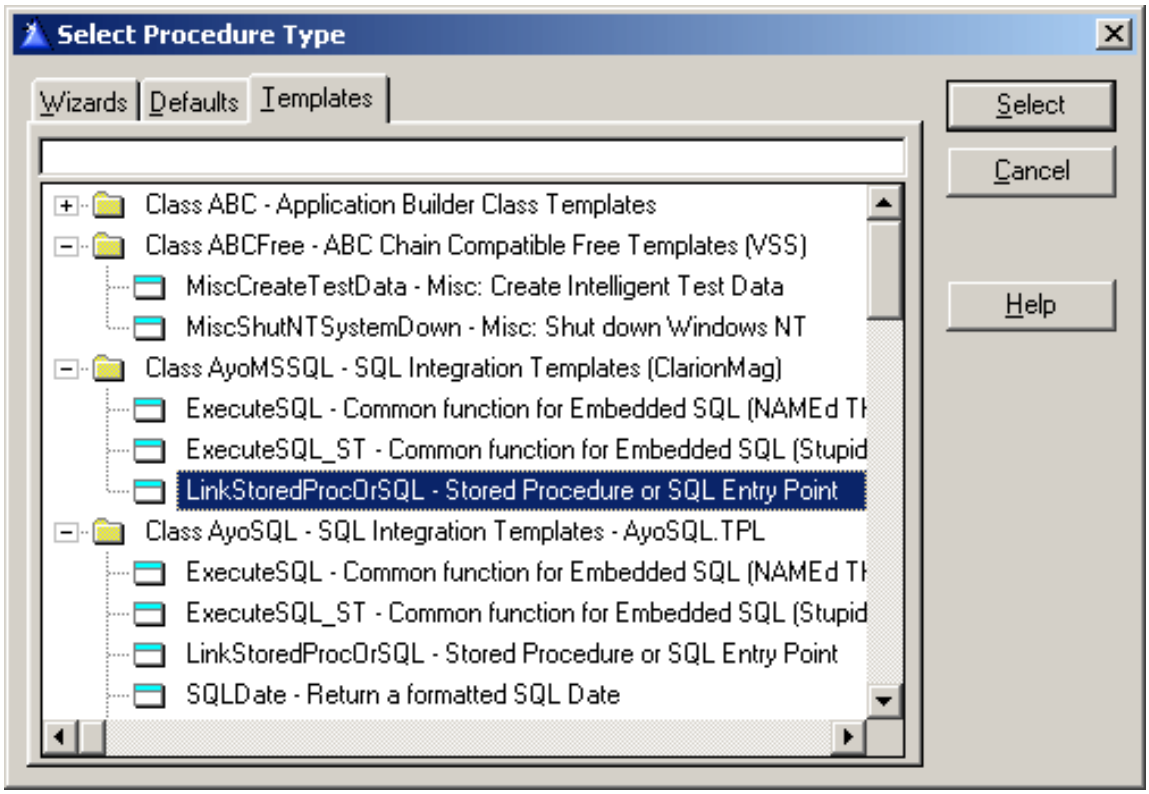


Figure 7. Adding Stored Procedure Template

This extension allows you to call and execute a stored procedure. In the full template version (which is not supplied as part of this article – email me for information on availability), you can also pass a SQL statement to SQL Server or even run a script file.

LinkStoredProcOrSQL Extension Template

The template prompts, as shown in Figure 8, are quite easy to follow.

Prototype: The prototype prompt follows the same standard as we are all used to except that you can enter the data type only. What this means is that a prototype like this is illegal:

(*DECIMAL pIncrease)

Rather, you have to enter:

(*DECIMAL)

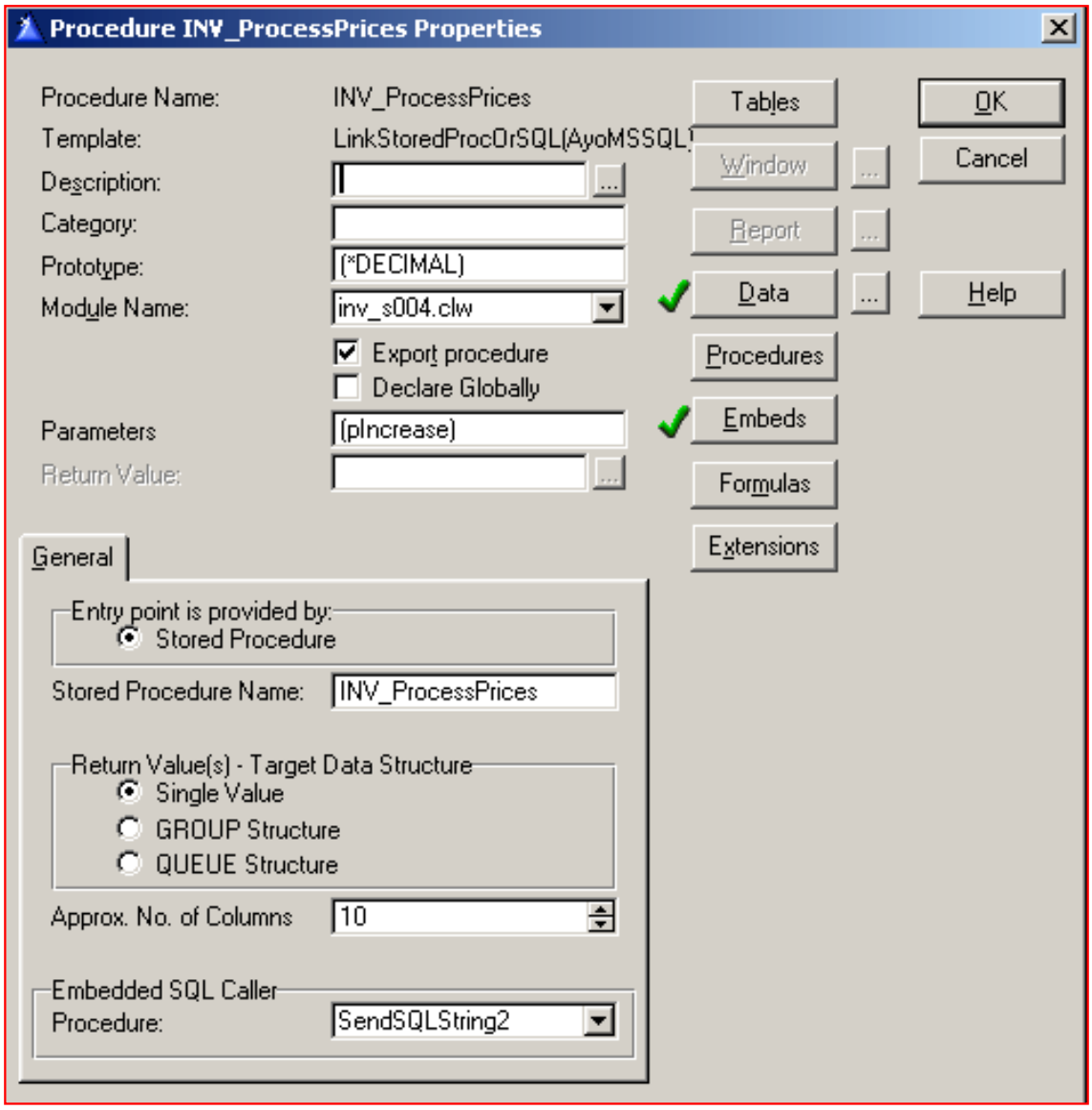


Figure 8. Stored procedure extension tem plate

The complete template set (not included as part of this article) allows you to return results from a stored procedure or SQL query into a `Queue`. This is how it works; if you want to return a result set from the called stored procedure or query, you have to pass a `TYPE`d `Queue` or `Group` as the last parameter in the prototype. For example, within an include file, or in your data section, you can define a `TYPE`d `Queue` this way:

```
myQueue_TYPE    QUEUE, TYPE
Field1            CSTRING(10)
Field2            CSTRING(20)
Field3            LONG
Field4            LONG
Field5            LONG
Field6            LONG
Field7            LONG
Field8            LONG
Field9            LONG
Field10            LONG
```

```
            END
```

In the case of the inventory example, I have to define this in `INV_APP_DATA.APP`.

My prototype then becomes:

```
(*DECIMAL, myQueue_TYPE)
```

I now define the actual queue inside INV_SQL.APP:

```
myQueue LIKE(myQueue_TYPE)
```

When this procedure is called from `INV_SQL.APP`, it will be called this way:

```
INV_ProcessPrices( PriceIncrease , myQueue )
```

After execution, `myQueue` will automatically get filled with the returned result set.

Parameters: The name of the parameter passed is entered here.

Entry Point: Normally, this contains two radio buttons. You can contact me if you want to purchase the template. The other one (not shown) allows you to type SQL query directly into the template, or even execute queries from query stored in an external file.

Stored Procedure Name: The default value generated here is the name of the procedure. If your procedure name matches the name of your stored procedure (as I suggested earlier), then you don't need to make any changes.

Return Value(s): Within the context of this article, you can leave the **Single Value** radio button selected as it is by default. The other selections **Queue**, **Group** are only useful if you are passing either a `Queue` or `Group` as the last parameter because you want values returned into them.

Approx. No. of Columns: This is an approximate number of columns to be returned by the stored procedure (just make sure you don't specify fewer columns than the stored procedure returns). Note that the number of fields described in your `TYPE`d definition for a `Queue` or `Group` should match the number indicated here.

Embedded SQL Caller: By default, the selection you made in the Global Template (see Step 5 in the previous article) will appear. To recap, the procedure(s) that appear here determine how the data access you are using is implemented. You can use the StupidTempTable Theory, or Smart Named Table.

After you have filled all the prompts, you can compile. The stored procedure is executed by code generated from the template.

Adding INV_ProcessPrices to (INV_SQL.APP)

Now that `INV_SQL_DATA.DLL` contains the call to the stored procedure, I need to create an external procedure that will allow me to call `INV_ProcessPrices` from `INV_SQL.APP`.

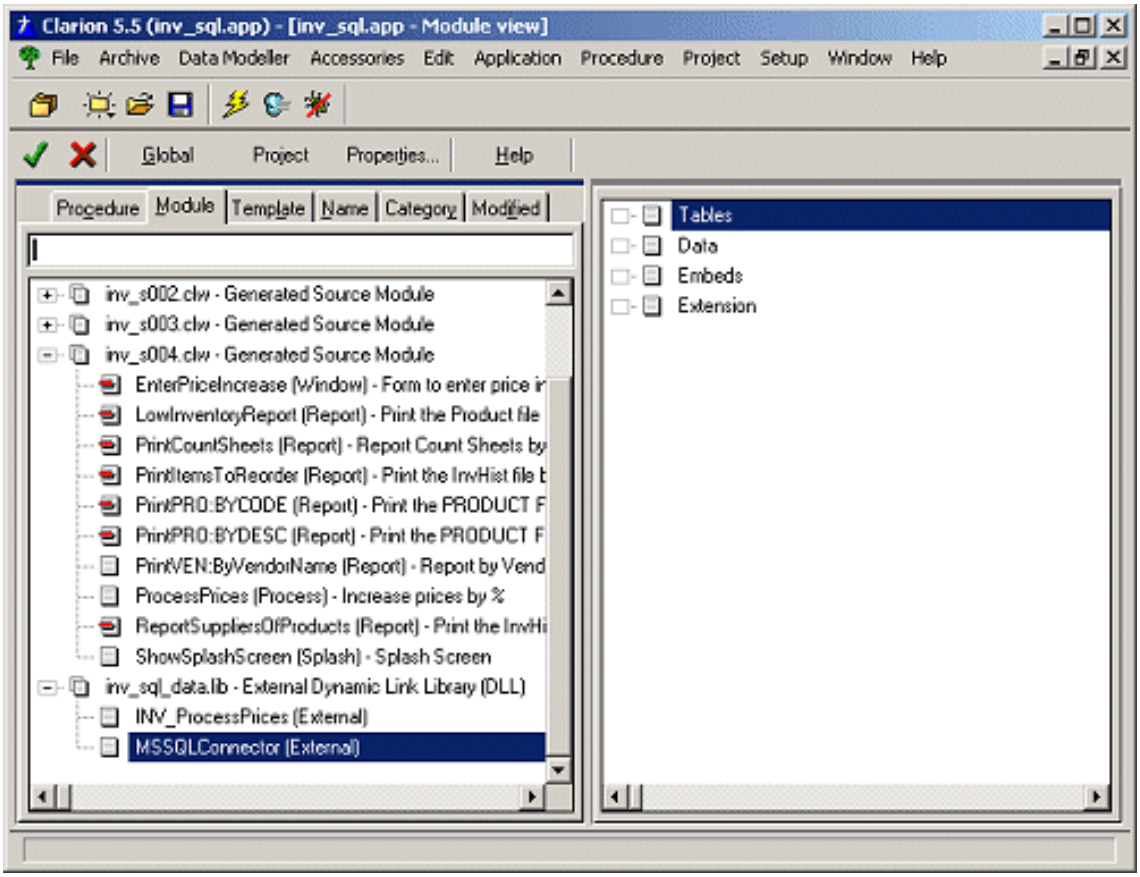


Figure 9. Adding INV_ProcessPrices

Now that I have added `INV_ProcessPrices` procedure to the application, I need to replace the `ProcessPricess` process procedure.

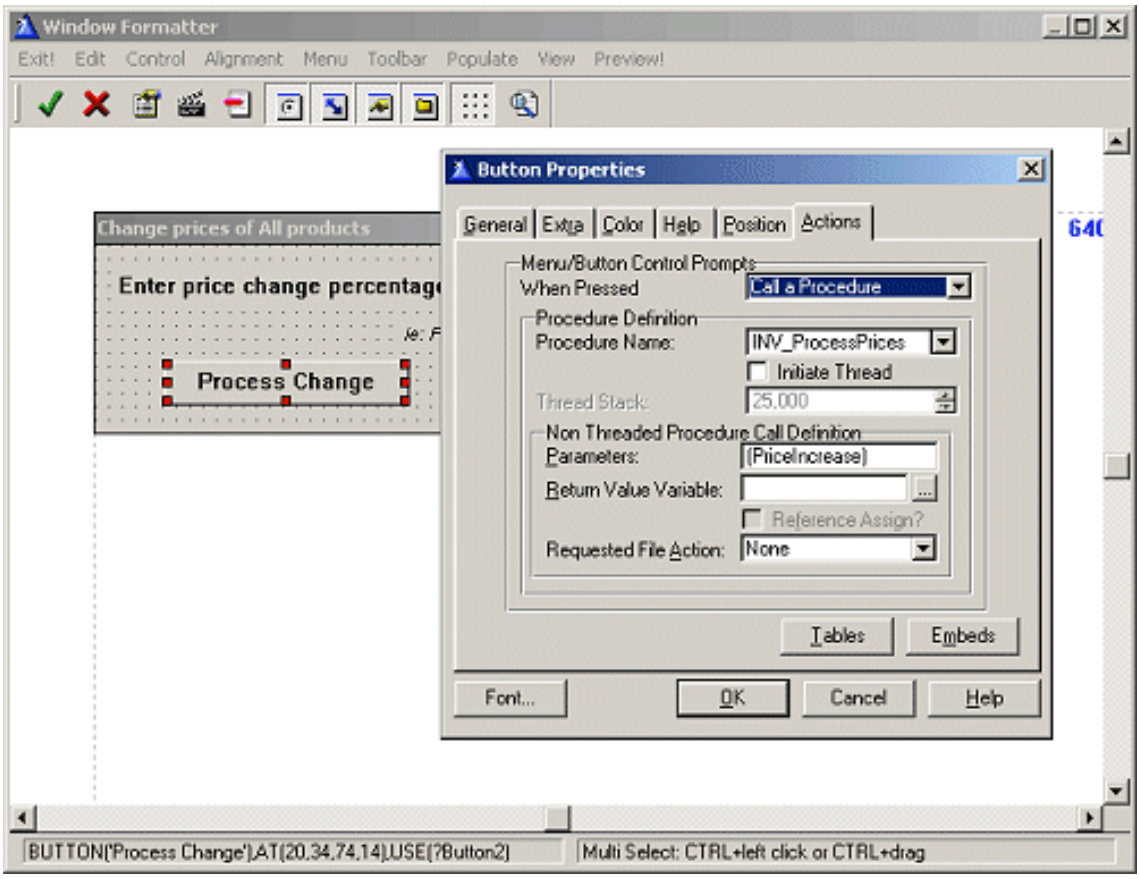


Figure 10. Calling New Price Increase Procedure

I'm now ready to save and compile the application. When I execute the **Price Increase** function from the menu, it will call the stored procedure.

Summary

I have shown you how to call a stored procedure using the supplied template. While this template has been crippled (contact me if you will like to have the full version) it still allows you to execute a stored procedure. I also mentioned two powerful tools that can aid you in debugging your SQL queries.

From this article, you can see that I have been able to isolate all my SQL coding to `INV_SQL_DATA.APP`. In fact, there isn't any embedded SQL code. I don't dispute the fact that sometimes this is not possible, especially when you want to be able to manipulate browses. However, coding your applications with external access and reuse of business logic in mind will save you a lot of maintenance headaches in the future.

Download the source

Ayo Ogundahunsi presently lives in Henderson, Nevada, about ten minutes from Las Vegas. He works for Impac Medical Systems

Inc., the leading company in cancer therapy software (written in Clarion). Impac has its headquarters in Mountain View, California. Ayo is married to Ayodola, and they have two boys, Darren and Joshua.

Reader Comments

Add a comment

A Class For The ASCIIng

by Konrad Byers

Published 2002-11-08

ASCII files are commonplace in programming. You may want to create a simple log file to help with debugging, or a text file for auditing purposes. Some command line utility programs let you pass complex command line parameters via an ASCII file. Custom search and replace programs need to be able to both read and write ASCII files.

All these tasks are generally more complicated in Clarion than in other languages because Clarion can only do ASCII file I/O via an ASCII file driver declared file. Declaring such a file can become quite a nuisance, and breaks my golden rule of never having to program the same thing twice. Enter the *any* ASCII file class.

The public methods the `AnyAsciiFileClass` are as follows:

```
CreateFile       PROCEDURE(STRING AsciiFileName, |
                   LONG OpenMode=ReadWrite+DenyWrite), BYTE, PROC
OpenFile         PROCEDURE(STRING AsciiFileName, |
                   LONG OpenMode=ReadWrite+DenyWrite), BYTE, PROC
OpenOrCreateFile PROCEDURE(STRING AsciiFileName, |
                   LONG OpenMode=ReadWrite+DenyWrite), BYTE, PROC
GetName          PROCEDURE(), STRING
Read             PROCEDURE(*CSTRING TextLine, |
                   ), BYTE, PROC
Read             PROCEDURE(*STRING TextLine, |
                   ), BYTE, PROC
Read             PROCEDURE(*PSTRING TextLine, |
                   ), BYTE, PROC
Replace          PROCEDURE(STRING TextLine), BYTE, PROC
Write            PROCEDURE(STRING TextLine), BYTE, PROC
IsOpen           PROCEDURE(), BYTE
GetReadPosition  PROCEDURE(), ULONG
SetReadPosition  PROCEDURE(ULONG ReadPosition)
```

```
GetWritePosition PROCEDURE(), ULONG
CloseFile        PROCEDURE()
SetAutoFlush     PROCEDURE(BYTE AutoFlush)
```

As you might expect, you may begin file processing with this class by calling either of the `CreateFile`, `OpenFile`, or `OpenOrCreateFile` methods. The `CreateFile` method will begin a new file, while `OpenFile` will open an existing one; `OpenOrCreate` will create a new file only when none already exists, otherwise it will open the existing file. These methods return `LEVEL:Benign` to indicate success, while any other value indicates failure.

An example

To use the class to process an input file, declare it as `InputFile AnyAsciiFileClass`. Next declare a variable to read a line of ASCII text. You will of course know more about the particular files you want to process than I do, but generally ASCII text file lines are less than 256 characters so I will declare a variable as `AsciiText CSTRING(256+1)` (+1 for the null terminator required for `CSTRING` variables).

Next up is simple I/O on ASCII files. The form of the `Read` method you will generally use will omit the second parameter and supply only the variable to receive the ASCII text. For the sake of explanation though, the read position parameter(s) can receive a value returned from the `GetReadPosition` method. This allows for a "re-read" of an ASCII text line at any previously read position. After doing so the current read position is just as it would be when this line was read the first time (ready to read the next subsequent line). The `SetReadPosition` method repositions the read position without actually reading anything until the next `Read` method is called. The multiple methods of setting the read position and then reading the text line at that position serve to provide some amount of flexibility.

The `Replace` method writes text to the file, as does the `Write` method (more about `Write` in a moment). `Replace` also updates the current read position. The `Replace` method has the same limitation as the Clarion ASCII database driver (because that's what it uses) that it cannot replace a line with a different number of bytes that was read without really fouling things up. However, this is only partially true of the `Replace` method. The `Replace` method jumps through some hoops to ensure that the replaced byte count *is* the same even when given a different size string to replace the existing text with. This will not be very useful when you are writing back more bytes, because the additional bytes will simply get clipped off. If you write back *fewer* bytes than were read however, the `Replace` method will space pad the replaced text line to the necessary and appropriate length. Ok, enough about the stuff you will probably never use.

The `Read` methods return `LEVEL:Benign` to indicate success and any other value to indicate failure, so a normal input file-processing loop would look like this:

```
IF InputFile.OpenFile('Some Existing File Name.txt') ~= LEVEL:Benign
   HALT(, 'Unable to open input file')
END
LOOP WHILE InputFile.Read(AsciiText) = LEVEL:Benign
   STOP(AsciiText)
END
STOP('Closing ' & InputFile.GetName())
InputFile.CloseFile()
```

The destructor of the class will actually close the file automatically if you neglect to, but it is simply good programming practice to undo everything on the way out that you do on the way. Accordingly, I call the `CloseFile` method explicitly.

The example also demonstrates the use of the `GetName` method, which simply returns the name of the currently open file.

The `Write` method appends text lines to the end of the file. The `SetAutoFlush` method has some impact on the performance and reliability of the `Write` method. When "auto flush" is on (set on with `SetAutoFlush(True)`), the file is opened before and closed after each `Write` method call. If you are using this class for a program debugging log file and your program is crashing then you really must turn this on. Otherwise the text may not be immediately written to the file, and in fact if the program crashes may never get written, leading to some very misleading conclusions about where the problem is. In general you can leave `AutoFlush` off (the default) for performance reasons.

The `GetWritePosition` method also relates to the `Write` method and can tell you the file pointer position where the next file write will or would occur. This is really only useful to move the read position to this point following a `Write` method call, or to determine the current size of the file.

All that is left is the `IsOpen` method, which you've probably guessed returns a Boolean indication of the files open status (i.e. is it open or is it not).

Below is an example of a program that reads from an input file and writes a modified form of the data to an output file.

```
 PROGRAM
 MAP
   INCLUDE('AnyAscii'),ONCE
 END

InputFile  AnyAsciiFileClass
OutputFile AnyAsciFileClass
AsciiText  CSTRING(256+1)

 CODE
```

```
IF InputFile.OpenFile('Some Existing File Name.txt') ~= LEVEL:Benign
  HALT(, 'Unable to open input file')
END
IF OutputFile.CreateFile('Newly Created File.txt') ~= LEVEL:Benign
  HALT(, 'unable to create output file')
END
LOOP WHILE InputFile.Read(AsciiText) = LEVEL:Benign
  OutputFile.Write(AsciiText & ' * modified output text line')
END
STOP('Finished - Closing files')
InputFile.CloseFile()
OutputFile.CloseFile()
```

The purpose of this class is not to make your code go really fast, it is to make your cod*ing* go really fast. Yes, using a hard coded ASCII file definition (or two, or three, depending on how many simultaneous streams are needed) would make for faster code but it is also slow to code that way and has its own problems. In a large program with multiple developers it may be next to impossible to determine what ASCII file definition and file name variable might be in use when and for what, and the situation will probably not remain in any given state indefinitely anyway.

Using this class eliminates these kinds of problems and risks by intentionally managing a single ASCII file declaration such that it may be used to implement multiple I/O streams on multiple files, one for each instance of the class. The class handles all the complexities of the file declarations reuse internally so your hands are freed up to write less problem prone code and write it faster. When you want to do something (such as read a file into a `QUEUE` for example) you can use the `AnyAsciiFileClass` to very easily take care of reading in the file, leaving you to concentrate on the larger problem as the file I/O aspects of the problem are implemented using pre-tested, canned code.

As you might now be thinking, reading the file into a `QUEUE` (and writing it back out again at some point) is a good idea. Doing so will allow random access to the text lines and eliminate the `Replace` method shortcomings inherited from the ASCII database driver.

Coming soon...

You can probably already guess what is coming… `AnyAsciiFileBufferClass` is born. This class is designed to be more or less interchangeable with the `AnyAsciiFileClass`. The primary differences between the two classes are that the `AnyAsciiFileBufferClass` makes a single pass through the file to *buffer* the entire contents in memory and will not access the file again until the file is closed with the `CloseFile` method. When the file is closed, any modified text lines will then be written back out to disk as necessary (i.e. if `Replace` or `Write` methods were called). Each class excels in the slightly different situations they are designed for. The `AnyAsciiFileClass` is best for single pass file reads or simple output file creating /appending, while the

`AnyAsciiFileBufferClass` excels at multiple pass reads or mass file writes. I'll walk through the `AnyAsciiFileBufferClass` next time.

Download the source

Konrad Byers is originally from Nova Scotia, Canada and has been living in Florida since 1995. He starting programming in Clarion with 2.1 for DOS, and is still an avid user of the Clarion C/C++ compiler. He is currently available for hire. A beta version of Konrad's MAPI-enabled eSoftAnywhere DSP & More digital signal processing software for shortwave and CB users and Ham Radio operators is now available for download.

Reader Comments

Add a comment

I'm the first to laud the efforts of class writers, but...

Data Structures and Algorithms Part XI - Binary Search Tree Indexing

by Alison Neal

Published 2002-11-08

Have you ever wanted to key index a text file so that you can fetch a single record directly? In this article I will show you how this can be done, using a Binary Search Tree.

I have covered several different forms of the Binary Search Tree in this series, but whichever form you choose, you only need to make small changes to use the tree as an index. You don't have to use a Binary Search Tree for indexing, however. You can index as easily using any number of structures, including the Clarion Queue, the List, or the Stack, but the Tree is one of the more efficient structures.

Indexing

To build a key or index on a file you need to designate which field in that file is going to be the key. In the sample application I have provided I have designated a randomly generated number as the primary key (although, since this is a text file, the data is stored as a string). At this stage I am limiting myself to numerical keys as new problems arise when using character or variable length string keys.

My file definition is as follows:

```
ValFile  FILE,DRIVER('ASCII'),NAME('Val.dat'),PRE(DAT),CREATE
Rec         RECORD
PrimKey     STRING(12)
DetailLine1 STRING(20)
DetailLine2 STRING(20)
            ..
```

Note that while the PrimKey field is defined in the file as a STRING, the test application stores this data as a ULONG, as the field contains a numeric value.

Having a file that I want to index, I then need to change the tree structure so that it can carry the key value and a pointer to the associated record on disk.

```
treeNode    CLASS,TYPE
PrimKey     ULONG
RecPos      LONG
ltree       &treeNode
rtree       &treeNode
            END
```

The PrimKey field in the treeNode class is going to correspond to the primKey in the file being indexed, and the RecPos field is going to contain a pointer to the record with that key value. To assign the pointer to a node means changing the parameters of the insert method, so that you pass both the primary key value (YourVal) and the record pointer (YourPos).

```
BTinsert PROCEDURE (ULONG YourVal,LONG YourPos),BOOL
```

With the creation of a new node, you just assign the RecPos at the same time as the PrimKey is assigned.

So, how do you get a pointer to a record? Well, Clarion has thankfully made this job very easy:

```
BIN.BTInsert(DAT:PrimKey,POINTER(ValFile))
```

This means that when my application starts, I can have it quickly scan through a file and build an index, so that records can thereafter be fetched directly.

How do I perform a fetch? Given a primary key value, I can use the find method to locate the node and get the pointer to the record:

```
binTree.BTfind          PROCEDURE(ULONG yourVal)
  CODE
  RETURN SELF.Bfind(SELF.root, yourVal)


binTree.Bfind  PROCEDURE(*treeNode t,ULONG yourVal)
  CODE
  IF t &= NULL THEN RETURN FALSE.
  IF t.primKey = yourVal
    SELF.curr &= t; RETURN t.recPos
  END
  IF t.primKey < yourVal
    RETURN SELF.Bfind(t.rtree, yourVal)
  ELSE
    RETURN SELF.Bfind(t.ltree, yourVal)
```

```
  END
```

If the `Find` function returns the pointer to the record, I can then read the record directly using the Clarion `GET` function:

```
GET(ValFile,BIN.BTFind(ValQ.PrimKey))
```

Indexing the ASCII file for reading is nothing difficult, however it would be useful to be able to update the index after inserting a record. The first thing I do is check to see that the primary key doesn't already exist in my tree index using the `Find` method, and if it does, gives an error that the duplicate key already exists. If the key doesn't exist I just add the record to the file, get the `POSITION()` of the new record, and then add the key to my Binary Tree index.

Take a closer look at the recursive `Find` method.

```
binTree.BTfind  PROCEDURE(ULONG yourVal)
    CODE
    RETURN SELF.Bfind(SELF.root, yourVal)
binTree.Bfind           PROCEDURE(*treeNode t,ULONG yourVal)
    CODE
    IF t &= NULL THEN RETURN FALSE.
    IF t.primKey = yourVal THEN SELF.curr &= t; RETURN t.recPos.
    IF t.primKey < yourVal
      RETURN SELF.Bfind(t.rtree, yourVal)
    ELSE
      RETURN SELF.Bfind(t.ltree, yourVal)
    END
```

For example, assume that the tree currently looks like Figure 1.

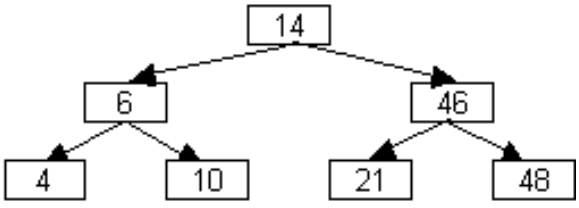


Figure 1. The tree to be searched

Now I want to find value 10, so I call the `BTFind` method passing the value that I want. `BTFind` calls the recursive `Bfind` method, passing both the root node and the value 10. As value 14 is greater than 10, and the coded rules of precedence are such that lower values are stored in the left sub tree and higher values are stored in the right sub tree, the first call to `Bfind` recurses left to the node containing value 6. This time, however, 10 is greater than 6 so another call is made to the right sub tree. On finding the correct value the method returns the stored pointer to the record . If the value being looked for did not exist in the tree then the

method would return `False`.

Summary

I've outlined a very simple form of key indexing an ASCII file, for direct fetches based on key value. There are limitations to this method however, the main one being that I am assuming that the data collection is so small that the key index will fit into memory. Other problems also arise if the key is character based rather than numerical. I will endeavor to address some of these problems in later articles.

Download the source

Alison Neal has been using Clarion since 2000, whilst working for Asset Information Systems (AIS) in Auckland, New Zealand. Some years ago (at the tender age of 19) Alison graduated from the Central Institute of Technology in Wellington, New Zealand with a major in Cobol. She also has a BA in English literature and has studied Computer Science, Philosophy and Information Systems. AIS is an independent division of Asset Forestry Ltd, and has a team of five programmers developing almost exclusively in Clarion. AIS also offers web (ClarioNET) and email services for the customer who needs everything. The company has many and varied customers bridging across a wide range of industries including Telecommunications, Forestry & Agriculture, Manufacturers, Military & Government, Legal & Financial, and Retail.

Reader Comments

Add a comment

Quick COM Using A Windows Script Component

by Wayne Price

Published 2002-11-13

Tap, tap, tap. Class, may I have your attention. Let's first make sure that you are all in the right room. What I'm going to be talking about today is the ability to access your Clarion code from another programming language, using COM. I am *not* talking about getting the latest COM widget to work in your Clarion program. That class is being taught down the hall by Jim Kane.

Alright then, now that that's out of the way, let's discuss a little bit more what you can do. In this article, you will learn how to use a Clarion function from VBScript. You will also learn how to adapt this to an ASP page. Let's get started.

The key is a Microsoft product that many of you may not even know about. It is the Microsoft Windows Script Component (WSC). You can get additional info at MSDN. This tool provides a bridge from your Clarion program to other languages that support a COM interface. The idea is that the ASP page, for example, communicates with the WSC component using COM, and the WSC component communicates with your Clarion program using `STDIN/OUT`, as shown in Figure 1.

Figure 1. Using WSC as a COM bridge

What is a Windows Script Component?

A Windows Script Component is a tool you can use to create a COM object that can be used on an ASP page, in VBScript, or with any other language that supports the COM interface.

You can get it free from Microsoft. While you are there you will also want to download the Windows Script Component Wizard.

A WSC file is simply an XML file that Windows recognizes as a COM object. The wizard just provides a few prompts that aids in the creation of this file. Once the file is created you have to register it by right-clicking on the file in Windows explorer and choosing **Register**.

Here is an example of a simple WSC:

```
<?xml version="1.0"?>
<component>
<registration
description="clartest"
progid="clartest.WSC"
version="1.00"
classid="{3f39648c-8393-4901-92c7-eb48deda604e}"
>
</registration>
<public>
<property name="LastName">
        <get/>
        <put/>
</property>
<method name="Initialize">
</method>
</public>
<implements type="ASP" id="ASP"/>
<script language="VBScript">
<![CDATA[
dim LastName
dim FirstName
function get_LastName()
get_LastName = LastName
end function
function put_LastName(NewValue)
LastName = newValue
end function
function Initialize()
end function
]]>
</script>
</component>
```

With a quick look you can see that the name of this object is `ClarTest` and it has a `LastName` property. The `get_LastName` and `put_LastName` methods allow you to set and read the `LastName` property. The `classid` is a unique id that is generated by the Wizard, and which can be used to identify the WSC.

STDIN/STDOUT

So how does all this relate to Clarion? As you may have noticed, there are three VBScript functions that you can use to communicate to a Clarion program. `WScript.Shell.Exec`, `StdIn.Write`, and `StdOut.ReadLine` allow you to run an executable and communicate to it through the `StdIn` and `StdOut` interface. If you code your Clarion program to watch for these communications, it can react to them. Look at this code taken from the WSC:

```
function Initialize()
Set WshShell = CreateObject("WScript.Shell")
Set oExec    = WshShell.Exec("c:\com4clar\com4clar.exe /COM")
end function
function Kill()
oExec.Terminate
end function
function getLastName()
oExec.StdIn.Write "getlastname()"
LastName = oExec.StdOut.ReadLine
end function
function pickLastName()
oExec.StdIn.Write "picklastname()"
LastName = oExec.StdOut.ReadLine
end function
```

The `WshShell.Exec` command runs the `com4clar` program that is written with Clarion. The `oExec.StdIn.Write` command sends a string to the Clarion program and the `oExec.StdOut.Readline` command gets the response from the Clarion program.

On the Clarion side, you use the `ReadFile` and `WriteFile` API functions to communicate back to the WSC. If you look in the IceTips FAQ you can find some Clarion code written by Frank Piscopo that demonstrates the use of `STDOUT`.

The Clarion program

In the following example, the program looks first to see if it was run with "`/COM`" as a parameter. This is just to keep someone from accidentally launching the program and leaving it running in the background. Next the application loops around waiting for any input from the WSC. When it receives a request, it passes the request through to the `EVALUATE` function and returns the result. This can be anything that `EVALUATE` can return. The Clarion program can also run a function, even one that has a Clarion window in it.

```
code
! Just in case the exe is accidentally run,
! it doesn't hang around forever.
if ~command('/COM') then return.
bind('NAM:LastName',NAM:LastName)
bind('getlastname',getlastname)
bind('picklastname',picklastname)
hStdIn = GetStdHandle(STD_INPUT_HANDLE)
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE)
```

```
  loop
    r = ReadFile(hStdIn, Address(InCString), 255, Address(lBytesRead), 0)
    if InCstring then
       OutCString = evaluate('clip('&InCString&')') & '<13,10>'
       r = WriteFile(hStdOut,OutCString,LEN(OutCString),lBytesWritten)
    end
    yield
  end
```

Putting it all together

You've seen all the individual parts, so now it's time to get the example to work.

1. Download the WSC tool kit from Microsoft here .
2. Download the example files. Be sure to save them in the `C:\Com4Clar` directory or you will have to change some of the source code.
3. Load and compile `Com4Clar.prj` in Clarion.
4. Register the WSC by navigating in FileExplorer to the `C:\Com4Clar` directory and right-clicking on the clartest.wsc and selecting **Register**.
5. Double click on the `c4ctest.vbs` file to run the VBScript.

An ASP version

I have also included an ASP file that will access this example Clarion program. You will have to adapt this to your server.

Here is the interesting part of the ASP file:

```
<%@ Language=VBScript %>
<object RUNAT=SERVER ID=clarTest
CLASSID="{3f39648c-8393-4901-92c7-eb48deda604e}">
</object>
<%clarTest.Initialize()%>
<HTML><BODY>
<H1>COM for Clarion Test</H1>
<%clarTest.FirstName="Wayne"%>
<%clarTest.getLastName()%>
First Name: <%=clarTest.FirstName%>
<b>This is set in the ASP Code.</b><br>
Last Name: <%=clarTest.LastName%>
<b>This is set in a clarion program that opens a
tps file and gets the first record.</b><br>
<%clarTest.Kill()%>
<br>
```

The ASP code is similar to the VBScript example but uses the `classid` to identify the WSC. Also be aware that as this is an ASP file you should not try to call a Clarion function that opens a window.

Conclusion

So what have you learned? Well, it is possible to open up your Clarion code to the rest of the world by using a Windows Script Component as a bridge to your Clarion code. Hopefully, this article will get you thinking of other ways to apply this technology.

Download the source

Reader Comments

Add a comment

Determining Gender With Clarion (Part 1)

by Geoff Robinson

Published 2002-11-14

A while ago I chanced across a humorous web site that generates complaint letters. Scott Pakin's automatic complaint-letter generator gets you to fill in a person's name and then spits out a funny "rambling but (I hope) grammatically correct complaint letter" about them. For those interested in the mechanics of such things the site gives the statistics of the context-free grammar used to generate the complaint letters.

Have a play with the complaint generator, but before emailing one of these complaints off to a friend (or enemy) perhaps you should read about an incident involving the automatic complaint-letter generator and the former mayor of Austin, Texas (19 August 2001).

The name Scott Pakin seemed to ring a bell with me, and eventually I remembered that someone of that name had written an AWK script, which determined someone's sex from their first name, that had been published in Computer Language magazine back in December 1991.

I emailed Scott letting him know I enjoyed his complaint generator and asking if he was the same guy who had written the Computer Language magazine article.

He responded:

```
Sent:         Saturday, 30 September 2000 9:58
Subject:      Re: complaint letter generator and AWK gender program
 Geoff,
 On Wed, 27 Sep 2000 17:25:19 +1100 you wrote:
 > I know you wrote it years ago but I've only just discovered your automatic
 > complaint-letter generator and thought I'd just say how much I enjoyed
 > it....
 Glad you like it!
 > Out of curiosity are you the same Scott Pakin who had an article in Dec 91
```

```
> Computer Language magazine with an AWK program to determine a person's
> gender based on their first (given) name??
Yes.  The little-known story that connects the two is this: I had written
the first version of the complaint-letter generator sometime in 1989.  I
thought it would be fun to write a script that would choose a random user
on my university's computer system and broadcast a brief complaint about
that person over Zephyr (an older, but more powerful, analogue of today's
instant-messaging systems).
The problem was that I needed to determine the gender of the person being
complained about.  At first, I just made my script always choose "male",
because males vastly outnumbered females at the university (Carnegie
Mellon).  However, the complaints sounded really stupid when the name
selected was obviously female.  So I made the script choose "female" if
the name ended in "a".  This worked a little bit better, but it was still far
from perfect.  I kept expanded the rules until I had an AWK script that
guessed first names fairly accurately.  I was so pleased with the outcome
  that I wrote up an article for Computer Language describing what I had
done, and, as you saw, they printed it -- my first "real" publication.

Anyway, thanks for writing,
-- Scott
```

I decided I would port the AWK script over to Clarion as another handy utility for my toolbox.

AWK is a marvellous little language by Aho, Kernighan and Weinberger (or perhaps that should be Aho, Weinberger and Kernighan if you are looking at the first letters…) that is ideal for processing text. It was originally written in 1977, and an enhanced version was released in 1985. AWK programs are often very small – just a couple of lines – because much of the work is done automatically for you by the language. I suppose you could compare it to a Clarion process (Sequential Record Processor) where you only have to write a little embed code as the templates generate the code to open files, read through the records etc. The difference is that with AWK that is all implicit with the language, and you are working at a very high level where you define patterns to match the input text, and actions to take when a given pattern is matched. Refer to "*The AWK Programming Language*" book (by the aforementioned Aho, Kernighan and Weinberger) for more details.

One of the nice things about AWK is that it has *associative arrays*. I first came across these in AWK in the late 80s, and then also used them in Paradox for Windows, which I used for Windows development prior to Clarion for Windows being released in 1994. Associative arrays were borrowed from the Snobol language, and are like normal arrays except that you can use any *string* as a subscript rather than just integers. So if you wanted to count how may times each word appeared in a text file you could use something like

```
count[word]++
```

to increment a count field for each word. If this is the first time the word has been found in the text a new entry is automatically added to the count array and initialised for you. Think for a

moment what you would have to do in Clarion….

When I moved to Clarion, boy did I miss those Associative Arrays! Whorf's hypothesis (see sidebar) suggests that man's thought is shaped by his tongue. This seems to be borne out by the fact that despite having missed associative arrays when first seriously using Clarion, these days I have so adjusted to the Clarion way of thinking that now I rarely miss them – I just work with what I have got. (Hey, but it would be a great feature to add in a future version of Clarion!)

Another feature of AWK is regular expressions. This feature was descended from the Unix tools grep and sed. More recently, Clarion introduced regular expressions (regex) with the `Match` function. I won't go into the details of regex here as you can read the on-line help on `Match` and, also read Carl Barnes' excellent article Using MATCH In Filters and Regular Expressions.

Scott Pakin had used a series of forty regular expressions as rules to determine the sex of a person when given their given name (got that?). As with many Clarion users, I'm a member of the Lazy Programmers Club, so rather than type in all the regular expressions I thought I'd get them from the net. I didn't find Scott's AWK script, but did find a Perl translation.

Perl is another interesting language. It was developed by Larry Wall who comes from a linguistic background, so Perl has a different slant from the majority of computer languages which are developed by computer scientists. Perl actually is descended in part from AWK, so it has all of AWK's capabilities plus much more (at the price of more complexity). The Perl Slogan is TMTOWTDI ("There's More Than One Way To Do It"), which is generally pronounced "tim-toady," but as the *Programming Perl* book dryly states, you can pronounce it however you like as, after all, TMTOWTDI. Looking at some of the code submitted for Clarion Challenges I think the slogan is equally applicable to Clarion - there are probably as many ways to do something as there are programmers to do it! By chance Perl is being used by Ron Schofield for the Open Clarion Project.

Looking at the Perl implementation I could see that the author had not fully understood the original intention. In the AWK

Whorf's Hypothesis

You have probably heard the story that Eskimos (Inuit?) have 50 or more words for "snow." The conclusion is that a person's thoughts are shaped and limited by his/her native language. This usually goes by the name of Whorf's hypothesis and is sometimes summarized as "Man's thought is shaped by his tongue."

Early in the twentieth century an anthropologist named Franz Boas made a remark to the effect that Eskimos had four unrelated root words for snow. The story goes that Benjamin Lee Whorf, a fire insurance inspector who had an interest in languages as a hobby, increased this to seven and just like the fish that got away this

version, the sex is initially set to "male" and then varied where appropriate to "female". Having read Scott's email above you will understand the historical reasons for that. If there is no rule that sets the sex, it is not because the function does not have any idea what the sex is, but rather it has no reason to think it is not male. Anyway the Perl implementation states:

> The original code assumed a default of male, and I am happy to contribute to the destruction of the oppressive Patriarchy by returning an UNDEF value if no rule triggers. Ha ha! Seriously, it'll be useful to know when gender() has no clue.

No matter how politically correct this might be, it is just plain wrong.

Perhaps I can explain it this way: Imagine there is some very structured society where life is very ordered and nonconformity is not tolerated. In this society there is a convention for names so that there can be no confusion as to the gender of someone granted that you know their name. The rules are:

1. All females names end in a single 'z'
2. All other names are of males, except where a name would end in 'z', in which case the last letter is always doubled: 'zz'.

As you can see, in this community a function to work out the gender is very simple. Let's suppose the rules are written as:

```
s = 'm' ! default to male
If last letter is 'z' then s = 'f'.
If last two letters are 'zz' then s = 'm'.
```

As you can see this will always be accurate. Someone with the name 'Abc' will be male, 'Xyz' will be female and 'Xyzz' will be male. But now someone comes along and says "I don't like to default to 'm' so I will say that there is no default". Suddenly all the male names which don't end in 'zz' are now 'undefined' and the function is far less useful.

Never mind, I had got hold of my regular expressions…

When looking at complex regexes, it is a good idea to break them down into their component parts so as to understand them more easily. When you write a regex you could think of it as defining the grammar for a very small language. The `Match` command compiles the grammar,

rapidly increased over time to urban legend status. Having said that, I did once see a list of "49 words for snow and ice from West Greenlandic" so there may be some truth in it after all… and I also read that some joker wanting a 50th entry added:

qayuqhak: *a snowdrift that happens to be shaped by the wind so that its profile resembles a duck's head*

and subsequently tests your string to see if some portion of it can be parsed as a sentence in your language. If it can, you then have a match!

Consider the regex for Allison and variations:

```
^All?[iy]((ss?)|z)on$
```

What this means is a string

1. starting with ‘A’
2. followed by either one or two ‘l’ (the question mark after the second ‘l’ means it is optional)
3. followed by either ‘i’ or ‘y’
4. followed by one or two ‘s’ or a ‘z’
5. and ending in ‘on’ (the $ indicates the ‘n’ must be the last letter)

At this stage I had problems getting things to work. I posted a message on one of the newsgroups, and Alexey came to the rescue with the information that, despite what the documentation says, Clarion uses non-standard curly braces{} rather than parentheses() for grouping. I thought this was a bug in Clarion’s implementation but Alexey demurred stating that it was a bug in the documentation! Different perspectives, obviously. I hope it will be corrected one day as using curly braces in strings are a real pain because you have to "double up" on the opening curly brace so that your braces do not match (if you’ll excuse the pun).

The corrected Allison example looks like this:

```
^All?[iy]{{{{ss?}|z}on$
```

I decided I was going to dig my heels in and still use my regexes with the elegant symmetry of the parentheses, and do a run-time conversion to curly braces. Then if Soft Velocity sees fit to fix the problem at some later date I can simply remove this step.

I wrote a little function called `VitRegMatch` that is passed the string to search, and the regex. It scans the string, replacing parentheses with curly braces, and then returns the result of the match:

```
VitRegMatch          FUNCTION (STRING p:str, STRING p:RE)
i  LONG
  CODE
! replace parentheses () with curly braces {}
    i = 1
    LOOP
       i = instring('(',p:RE,1,i)
       if i
           p:RE[i] = '{{' ! is only one byte
       else
```

```
            break
        end
    END
    i = 1
    LOOP
        i = instring(')',p:RE,1,i)
        if i
            p:RE[i] = '}'
        else
            break
        end
    END
    return(match(p:Str,p:RE,Match:Regular))
```

As I said before TMTOWTDI, and I could have simply looped through character by character manually rather than using `instring()` – I'll revisit that later. This implementation though is interesting in that I don't have to check for the end of the string; that is handled by `instring()` returning 0 when there are no more matches. This is somewhat reminiscent of the sentinel trick when searching, where you append what you are searching for to the end of the string that is being searched. That way you are *guaranteed* a match and so do not have to check whether you are at the end of the string each time when looping. When you find a match all you have to do is see if your matching string is within the bounds of original string.

I should mention that I usually add the letters "Vit" to the start of my function names just to differentiate them from Clarion functions and those from third parties. Vit is just short for Vitesse, which is my company name.

With my `VitRegMatch` in hand my first version of the gender guessing function looked like this:

```
VitGetGender1        FUNCTION (STRING p:Name)
s   STRING('m')   ! sex defaults to male...
  CODE
    if ~p:Name then return('?').  ! no name passed...

    p:Name = lower(p:Name)
    p:Name[1] = upper(p:Name[1])  ! capitalise name
    if VitRegMatch(p:name, '^.*[aeiy]$')
      then s = 'f'. ! most names ending in a/e/i/y are female
    if VitRegMatch(p:name, '^All?[iy]((ss?)|z)on$')
      then s = 'f'. ! Allison and variations
    if VitRegMatch(p:name, '^.*een$')
      then s = 'f'. ! Cathleen, Eileen, Maureen
    if VitRegMatch(p:name, '^[^S].*r[rv]e?y?$')
      then s = 'm'. ! Barry, Larry, Perry
    if VitRegMatch(p:name, '^[^G].*v[ei]$')
      then s = 'm'. ! Clive, Dave, Steve
    if VitRegMatch(p:name, '^[^BD].*(b[iy]|y|via)nn?$')
      then s = 'f'. ! Carolyn, Gwendolyn, Vivian
    if VitRegMatch(p:name, '^[^AJKLMNP][^o][^eit]*([glrsw]ey|lie)$')
```

```
  then s = 'm'. ! Dewey, Stanley, Wesley
if VitRegMatch(p:name, '^[^GKSW].*(th|lv)(e[rt])?$')
  then s = 'f'. ! Heather, Ruth, Velvet
if VitRegMatch(p:name, '^[CGJWZ][^o][^dnt]*y$')
  then s = 'm'. ! Gregory, Jeremy, Zachary
if VitRegMatch(p:name, '^.*[Rlr][abo]y$')
  then s = 'm'. ! Leroy, Murray, Roy
if VitRegMatch(p:name, '^[AEHJL].*il.*$')
  then s = 'f'. ! Abigail, Jill, Lillian
if VitRegMatch(p:name, '^.*[Jj](o|o?[ae]a?n.*)$')
  then s = 'f'. ! Janet, Jennifer, Joan
if VitRegMatch(p:name, '^.*[GRguw][ae]y?ne$')
  then s = 'm'. ! Duane, Eugene, Rene
if VitRegMatch(p:name, '^[FLM].*ur(.*[^eotuy])?$')
  then s = 'f'. ! Fleur, Lauren, Muriel
if VitRegMatch(p:name, '^[CLMQTV].*[^dl][in]c.*[ey]$')
  then s = 'm'. ! Lance, Quincy, Vince
if VitRegMatch(p:name, '^M[aei]r[^tv].*([^cklnos]|([^o]n))$')
  then s = 'f'. ! Margaret, Marylou, Miri;
if VitRegMatch(p:name, '^.*[ay][dl]e$')
  then s = 'm'. ! Clyde, Kyle, Pascale
if VitRegMatch(p:name, '^[^o]*ke$')
  then s = 'm'. ! Blake, Luke, Mi;
if VitRegMatch(p:name, '^[CKS]h?(ar[^lst]|ry).+$')
  then s = 'f'. ! Carol, Karen, Shar;
if VitRegMatch(p:name, '^[PR]e?a([^dfju]|qu)*[lm]$')
  then s = 'f'. ! Pam, Pearl, Rachel
if VitRegMatch(p:name, '^.*[Aa]nn.*$')
  then s = 'f'. ! Annacarol, Leann, Ruthann
if VitRegMatch(p:name, '^.*[^cio]ag?h$')
  then s = 'f'. ! Deborah, Leah, Sarah
if VitRegMatch(p:name, '^[^EK].*[grsz]h?an(ces)?$')
  then s = 'f'. ! Frances, Megan, Susan
if VitRegMatch(p:name, '^[^P]*([Hh]e|[Ee][lt])[^s]*[ey].*[^t]$')
  then s = 'f'. ! Ethel, Helen, Gretchen
if VitRegMatch(p:name, '^[^EL].*o(rg?|sh?)?(e|ua)$')
  then s = 'm'. ! George, Joshua, Theodore
if VitRegMatch(p:name, '^[DP][eo]?[lr].*s$')
  then s = 'f'. ! Delores, Doris, Precious
if VitRegMatch(p:name, '^[^JPSWZ].*[denor]n.*y$')
  then s = 'm'. ! Anthony, Henry, Rodney
if VitRegMatch(p:name, '^K[^v]*i.*[mns]$')
  then s = 'f'. ! Karin, Kim, Kristin
if VitRegMatch(p:name, '^Br[aou][cd].*[ey]$')
  then s = 'm'. ! Bradley, Brady, Bruce
if VitRegMatch(p:name, '^[ACGK].*[deinx][^aor]s$')
  then s = 'f'. ! Agnes, Alexis, Glynis
if VitRegMatch(p:name, '^[ILW][aeg][^ir]*e$')
  then s = 'm'. ! Ignace, Lee, Wallace
if VitRegMatch(p:name, '^[^AGW][iu][gl].*[drt]$')
  then s = 'f'. ! Juliet, Mildred, Millicent
if VitRegMatch(p:name, '^[ABEIUY][euz]?[blr][aeiy]$')
  then s = 'm'. ! Ari, Bela, Ira
if VitRegMatch(p:name, '^[EGILP][^eu]*i[ds]$')
  then s = 'f'. ! Iris, Lois, Phyllis
if VitRegMatch(p:name, '^[ART][^r]*[dhn]e?y$')
```

```
    then s = 'm'. ! Randy, Timothy, Tony
  if VitRegMatch(p:name, '^[BHL].*i.*[rtxz]$')
    then s = 'f'. ! Beatriz, Bridget, Harriet
  if VitRegMatch(p:name, '^.*oi?[mn]e$')
    then s = 'm'. ! Antoine, Jerome, Tyrone
  if VitRegMatch(p:name, '^D.*[mnw].*[iy]$')
    then s = 'm'. ! Danny, Demetri, Dondi
  if VitRegMatch(p:name, '^[^BG](e[rst]|ha)[^il]*e$')
    then s = 'm'. ! Pete, Serge, Shane
  if VitRegMatch(p:name, '^[ADFGIM][^r]*([bg]e[lr]|il|wn)$')
    then s = 'f'. ! Angel, Gail, Isabel
  return(s)
```

To be honest it wasn't quite my first version, as originally I called it `VitGetSex` until my partner, noticing this, asked what the heck I was working on! So `VitGetGender` it is – and this article is titled *Determining Gender With Clarion* and not the easily misconstrued *Using Clarion To Get Sex!*

It was with mild amusement that I later noticed that Scott Pakin had, at around the same time, "Modified [the] form interface and generated URLs to use the word 'gender' instead of a particular, 3-letter synonym [so that] users trapped behind censorware programs and proxies can now use the automatic complaint-letter generator."

Moving right along…

`Match()` does allow you to be case insensitive (`Match:NoCase`) but I decided it was likely to be less overhead to capitalize the name (which is what the regexes expect) and just do a case sensitive match.

Note that all of the regexes as written start with `^` and end with `$`. The caret `^` (or *anchor* as it is sometimes referred to in regexes) indicates the start of the string, while the `$` indicates the end of the string. Hence the whole string has been defined, and while this is not really necessary in all cases (eg. ^.*een$ could have simply been written as een$ meaning *ending in 'een'*) it again lends a certain symmetry.

That's all for this installment. Next week I'll show how to move the regexes into the data area, and I'll discuss the results of my testing.

Download the source

Geoff Robinson lives near the beach in Melbourne, Australia, and is an active member of the local Clarion User Group. His company, Vitesse Information Systems, specializes in software for local government. Geoff was impressed by Clarion back in the DOS days and grabbed the early betas of Clarion for Windows when they first became available; he has been using Clarion as his

primary development environment ever since. When not in front of a computer Geoff enjoys listening to music, singing bass in a local choir, and spending time with his three young children.

Reader Comments

Add a comment

Match:NoCase does not apply to character sets e.g....
Thanks Carl - lucky I did it that way then! 8-) I...
It was some patterns like...
Thanks again Carl for your sage comments. As it turns...

SQL Data Types Comparison

by David Harms

Published 2002-11-15

The following table summarizes data type differences between three popular open source databases, Firebird, MySQL, and PostgreSQL, and Microsoft's SQL Server. Please note that is a guideline rather than a definitive reference – I don't have personal experience with all of these databases.

As you can see from the table, there is really very little standardization on either type names or specifications. In fact, there is only *one* data type you can use identically across all four servers, and that is the `BIGINT` type. `INT` or `INTEGER` types come a close second, but MySQL allows you to use an unsigned form which changes the range of values.

In many cases similar data types are available, but have different names. If you're storing a BLOB, depending on the size of the BLOB and the server, you might call that data type `BLOB`, `BYTEA`, `BINARY`, `TINYBLOB`, `MEDIUMBLOB`, `LONGBLOB`, or `IMAGE`. String data doesn't fare that much better. Although all four servers have a `CHAR` data type, each has a different size limit. In MySQL, it's 255 characters; MS SQL, 8000 characters; Firebird, 32767 characters; PostgreSQL, approximately 1 gigabyte. Variable length strings (`VARCHARs`) have the same limits in each database as `CHAR`.

Each server also has its own implementation for autoincrementing unique values; for instance, MySQL's `AUTO_INCREMENT` attribute on keys won't do you any good in PostgreSQL, where you need to make use of a Generator object.

Things get really hairy when you start dealing with fields that are a combination of a date and a time. You'll want either a `DATETIME` field or a `TIMESTAMP` field. But be careful! MySQL and SQL Server have `DATETIME` types which fit this description. In Firebird and PostgreSQL

you have to use a TIMESTAMP. PostgreSQL's TIMESTAMP, however, can include a GMT offset, something not available in any of the other TIMESTAMP or DATETIME types. And don't think you can use the TIMESTAMP field in MySQL or SQL Server the same way as in Firebird or PostgreSQL. A MySQL TIMESTAMP field is set to the current date/time whenever you update the record, so you can't use it to store an arbitrary value; SQL Server is similar, although its value is also guaranteed to be unique across the entire database. And if you're converting between databases, keep in mind that there may be incompatibilities between the default date/time formatting.

Even simple DATE or TIME fields can present problems. Again, PostgreSQL TIME fields can store time zone offsets from GMT, which no other TIME type listed here supports. And SQL Server doesn't have DATE or TIME types, just DATETIME.

The moral of the story? As Mike Gorman has said, there really is no such thing as an SQL standard. Unless you know you'll never have to support more than one SQL server, try to stick with the more commonly supported data types. Be wary of arrays, boolean and bit fields, MySQL's ENUMs, and Postgress's specialized and user-defined data types.

If you have any corrections or comments, please post them at the end of this article.

Type	Firebird	MySQL	PostgreSQL	MS SQL Server
BIGINT	-9223372036854775808 to 9223372036854775807	-9223372036854775808 to 9223372036854775807	-9223372036854775808 to 9223372036854775807	-9223372036854775808 to 9223372036854775807
BIGSERIAL	use a GENERATOR object	use AUTO_INCREMENT on primary key	auto-incrementing integer, 1 to 9223372036854775807	use IDENTITY
BINARY	see BLOB	see BLOB, TINYBLOB, MEDIUMBLOB, LONGBLOB	see BYTEA	Binary data, max 8000 bytes,
BIT	n/a	see TINYINT	Bit masks	0 or 1

BIT VARYING	n/a	n/a	Variable length bit masks	n/a
BLOB	Segment size limited to 64K, no limit on BLOB size	BLOB or Text up to 65535 bytes	see BYTEA	see IMAGE, TEXT, NTEXT
BOOLEAN	n/a	(BOOL) see TINYINT	True values include: TRUE, 't', 'true', 'y', 'yes', '1' – false values include FALSE, 'f', 'false', 'n', 'no', '0'	see BIT
BYTEA	see BLOB	see BLOB	Binary string, no specific limit on size	see BLOB
CHAR(n)	1 to 32767 characters	0to 255 characters	approx 1 GB limit	1 to 8000 characters
DATE	8 bytes, 1 Jan 100 to 29 Feb 32768	1000-01-01 to 9999-12-31	4713 BC to AD 1465001	see DATETIME
DATETIME	see TIMESTAMP	1000-01-01 00:00:00 to 9999-12-31 23:59:59	see TIMESTAMP	January 1, 1753, to December 31, 9999, accuracy 3.33 milliseconds

DECIMAL (precision,scale)	2, 4 or 8 bytes, precision=1-18, scale-0-18. Scale is the decimal places, must be <= precision.	-1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+30	user-specified precision, exact, no limit	-10^38 +1 to 10^38 –1
DOUBLE PRECISION	8 bytes, range 2.225 x 10^{-308} to 1.797 x 10^{308}	-1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+30	8 bytes, up to 15 decimal places precision	see MONEY
ENUM	n/a	An enumeration of allowed values (similar to a CHECK constraint)	n/a	n/a
FLOAT	4 bytes, range 1.175 x 10^{-38} to 3.402 x 10^{38}	-3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38	see DOUBLE PRECISION	-1.79E + 308 to -2.23E to 308, 0 and 2.23E + 308 to 1.79E + 308
IMAGE	see BLOB	see BLOB	see BLOB	Variable length binary data, max 2GB
INT INTEGER	4 bytes, range –2,147,483,648 to 2,147,483,647	-2147483648 to 2147483647 or 0 to 4294967295	-2147483648 to +2147483647	-2,147,483,648 to 2,147,483,647

INTERVAL (between two TIME or TIMESTAMP values)	n/a	n/a	12 bytes, resolution one microsecond, -178000000 to 178000000 years	n/a
LONGBLOB LONGTEXT	see BLOB	BLOB or Text up to 4 GB	see BLOB	see BLOB
MEDIUMBLOB MEDIUMTEXT	see BLOB	BLOB or Text up to 16777215 bytes	see BLOB	see BLOB
MONEY	see DECIMAL	see DECIMAL	Fixed precision (two decimal places), range -21474836.48 to +21474836.47	-922,337,203,685,477.5808) to +922,337,203,685,477.5807
NCHAR	see CHAR	see CHAR	see CHAR	Unicode string, max 4000 characters
NTEXT	see BLOB	see BLOB	see TEXT	Unicode text, max 1 GB
NVARCHAR	see VARCHAR	see VARCHAR	see VARCHAR	Unicode string, max 4000 characters
NUMERIC (precision,scale) (usually eqiuivalent to decimal type)	2, 4 or 8 bytes, precision=1-18, scale-0-18. Scale is the decimal places, must be <= precision.	-1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+30	user-specified precision, exact, no limit	-10^38 +1 to 10^38 –1

REAL	see FLOAT	-1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+30	4 bytes, up to 6 decimal places precision, floating point	-3.40E+38 to -1.18E-38, 0 and 1.18E-38 to 3.40E + 38
SERIAL	use a GENERATOR object	use AUTOINCREMENT primary key	Auto-incrementing integer, 1 to 2147483647	use IDENTITY
SET	n/a	A string that can have zero or more values from the allowed list.	n/a	n/a
SMALLDATETIME	see TIMESTAMP	see DATETIME	see TIMESTAMP	January 1, 1900, to June 6, 2079, accuracy 1 minute
SMALLINT	–32,768 to 32,767	-32768 to 32767 or 0 to 65535	-32768 to +32767	–32,768 to 32,767
SMALLMONEY	n/a	n/a	n/a	-214,748.3648 to +214,748.3647
TEXT	n/a	n/a	Approx 1 GB limit	2 GB limit
TIME	8 bytes, 0:00 AM-23:59.9999 PM	-838:59:59 to 838:59:59	range 00:00:00.00+12 to 23:59:59.99-12 (shown with optional timezone notation)	see DATETIME

TIMESTAMP	Combination of date and time	1970-01-01 00:00:00 to sometime in the year 2037, automatically set to the date/time of the most recent update of the row	8 bytes, can include time zone, range 4713 BC to AD 1465001	A database-wide unique number that gets updated every time a row gets updated
TINYBLOB TINYTEXT	see BLOB	Text or BLOB up to 255 bytes	see BLOB	see BLOB
TINYINT	see SMALLINT	-128-127, or 0-255	see SMALLINT	0-255
UNIQUEIDENTIFIER	n/a	n/a	n/a	Globally unique identifier (GUID)
VARBINARY	see MEDIUMBLOB	see BLOB	see BLOB	Max 8000 bytes
VARCHAR(n)	1 to 32,765 bytes	0 to 255 bytes	Approx 1 GB limit	1 to 8000 characters
YEAR	n/a	A year in 2- or 4-digit format	n/a	n/a
Arrays	All datatypes except BLOBs	n/a	All built-in or user-defined data types	n/a
Geometric types	n/a	n/a	PostgreSQL includes a number of geometric data types such as line, point, lseg, box, path, polygon, and circle.	n/a

Network address types	n/a	n/a	PostgreSQL includes the following network address data types: cidr, inet, and macaddr	n/a
User defined types	n/a	n/a	You can create additional types with the CREATE TYPE command	n/a

---

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995). His most recent book is JSP, Servlets, and MySQL, published by HungryMinds Inc. (2001).

Reader Comments

Add a comment

Clarion Magazine

Clarion News

Published 2001-11-21

Business Rules Engine Beta Testers Required.

Quantum Dynamics International is looking for beta testers for eXpert Rules, a template-based command language. The Rule based programming paradigm is patentable so all testers will be required to sign an NDA and not disclose examples programs created or the scripts they create. The Rules processor has been create in Clarion with a customisable front end in Visual Basic. Features include: Window management system; Window Control population command set; Edit In Place command Set; Report Command Set; Database command set; Data Dictionary command set; List command set; View command set; Data Sheet Cube command set; Field Object, field array and History command set; VBA embeded code; COM Objects via VB callbacks; VBA callback support system from the Rules Processor.

Posted Friday, November 29, 2002

QuickBooks SDK 2.0

The QuickBooks SDK 2.0 was released Tuesday 11/13/2002. It is freely available for download from Intuit. Signing up for the community developer level membership is free and gives you access to the Knowledge Base and other SDK information.

Posted Friday, November 29, 2002

Hotfix For ImageEx2 Beta 2

A hotfix for ImageEx2 beta 2 is now available. This fixes a problem with TWAIN scanning: The first call to the AcquireFromTwain method of the ImageExBitmapClass would work correctly, but subsequent calls would fail. Extract IMEXCORE.DLL to your Clarion\Bin directory.

Posted Friday, November 29, 2002

Helpmaker Website Change

AccViz HelpMaker (freeware) is now available on the AccViz web site.

Posted Friday, November 29, 2002

MySQL Browse Class Updated

Dan Pressnell has updated his class for page loaded MySQL browses. Improvements include: Smaller record sets for faster performance; List box refreshes when window is resized; Scrolling and paging are now more accurate when you have a non-unique key selected for the primary file, but you have "Additional sort fields" that make each record unique.

Posted Friday, November 29, 2002

ImageEx2 Beta 2

The second beta version of ImageEx 2 is now available. Changes include: Added methods to support "stippled" lines; Added the Bezier method which draws bezier curves; Added the Roll method to scroll images; Added the PngSaver class to save images in PNG format; Modified the drawing functions to accept single-precision floating point parameters as coordinates (for use with antialiasing); Added the SetFont and GetTextExtent methods for better control of text output; Improved the AcquireFromTwain method for more control on TWAIN scanning; Added the HslToRgb color conversion function; Added several help topics with general info. A new demo is available, which includes a small space shuttle game which shows how to do collision detection using the alpha channel of two images, as well as a demo showing how to draw gradient lines using the new StippleLine method. Demo source is included with ImageEx 2. The cost is $149 US while in beta. The gold release (expected soon) will be $199 US.

Posted Friday, November 29, 2002

WebUpdate Template Corrected

The RUpdate template zip (Web update for applications) file had, for some reason, become corrupted. Clarionshop has been updated with the corrected file.

Posted Monday, November 25, 2002

MySQL Help Document

Vernon Jay Godwin has put together a document with information on common problems encountered when using MySQL with Clarion together.

Posted Monday, November 25, 2002

List & Label v9 Demo

A demo of List & Label v9 is now available. This new version includes: New export formats to Excel, Text and Tiff; Completely revamped layout designer (which includes a lot of requests from Clarion users); New OLE facility - you can now embed OLE objects such as Excel charts, Word documents, HTML documents etc into your reports; Fax from reports; Renewed Previewer with thumbnails; Switchable XP look and feel to the designer; New dockable toolboxes; New properties such as the Clarion equivalent of blank when zero pictures; Ability to 'lock' objects to prevent them being accidentally selected and moved. The German version

has been released and the new English version is due out at the beginning of December. Orders taken before the release receive a $50 discount. There is also a new discounted version of List & Label. See the website for details.

Posted Monday, November 25, 2002

IconsXP Specialty Groups

Gitano has release its IconsXP Specialy Groups. Each group contains images for a specific subject. These are not actually icons, but high resolution images that can be resized using any standard program such a PaintShop Pro (c). You can resize to use on a tool bar, splash screen, web sites, etc. Each set is distributed in two formats, one that will contain all layers in the image, preserving the transparency, so that you can modify, and a flat image with a shadow that is ready to use as is.

Posted Monday, November 25, 2002

Date Handling Class

Fiscal Software has a free class available that handles various date/time-related functions, including day names and numbers, elapsed time, last given day of the week in the month, etc.

Posted Monday, November 25, 2002

CapeSoft Updates

The following products have been updated since the last newsletter: NetTalk 2.5 beta; Insight Graphing 1.0 beta 7j; CapeSoft Draw 1.0 beta 11; HyperActive 1.7e; File Explorer 2.5c ; CapeSoft MessageBox 1.6g; Special Agent 1.42.

Posted Monday, November 25, 2002

File Manager 3 Beta Progressing

The CapeSoft File Manager 3 beta is progressing. Both MSSQL and ORACLE driver support has been significantly improved, and the ODBC development is coming along well. You can now use the ODBC driver with FM3 for MSSQL databases. The last month has seen major improvements in file change detection, structure comparison, key and index support, and datatype matching. MySQL support is under development. FM3 is US$149 while in beta, and will be US$199 in gold release. For current FM2 users there is a cross-grade special of US$99.

Posted Monday, November 25, 2002

Office Inside

CapeSoft's Office Inside uses Thinkware's Clarion COM technology to make it easy to implement Office functionality into your apps. Tasks like spell-checking, generating editable Word documents, and many other MS Office tasks can be implemented in minutes. CapeSoft Office Inside is selling for US$199 while in beta testing, but the price will increase to US$299 with the Gold release.

Posted Monday, November 25, 2002

CapeSoft's Replicate Beta

CapeSoft's Replicate provides an automatic, driver independent, file-version independent, mechanism for replicating data in two or more databases. Replicate logs your changes, adds and deletes and then using a transport manager of your choice, exports the changes to the other sites, where the changes, adds and deletes are imported to that data set. This is all done completely automatically without your users having to do anything. Replicate can be used in a LAN environment, or for asynchronous replication in an online or offline environment. Replicate is selling for US$349 while in beta testing, but the price will increase to US$499 with the gold release.

Posted Monday, November 25, 2002

MySQL Page Loaded ABC Browses

Dan Pressnell has a class available which makes MySQL practical to use with large result sets. This class is a drived ABC browse class which adds the LIMIT clause to the SELECT statements. This lets you keep all the browse familiarity you're accustomed to. One side effect is that scrolling will be a little slower, because rather than just FETCHING the next set of rows on, for example, a page down, a new query will have to be sent to the server. Follow the instructions in MySQLBr.inc.

Posted Monday, November 25, 2002

CPCS Previewer Enhancements

The CPCS Previewer for v5.16 (C5) and v5.57h (C55) has been enhanced with the following functionalities: Stay-After-Print Toggle button added - when depressed, allows you to stay in the previewer after printing the report; Additional add-on buttons are now toggle buttons - the action associated with each button will only be performed if the button is in a "depressed" state upon leaving the preview. These builds are free to all currently registered users of v5.16 and/or v5.57h (install codes are the same as for the initial release of these versions).

Posted Monday, November 25, 2002

HTML Designer Version 1.03 Beta

HTML Designer version 1.03 Beta is now available. Changes include: Web Update facility incorporated - all future updates will be done through the Help/About window in HTML Designer; Multi Help projects are now available for each application; Various bug fixes and help file updates.

Posted Monday, November 25, 2002

Search Engine Profile Exchange Update

Encourager Software has created an centralized information resource for individuals or companies that need to submit their web sites and/or software programs to the various search engines. The File and Search Engine Discussion is collection of advice to help you focus on

marketing techniques for search engine optimization and placement (software author advice also). Online and RTF version (in SearchPS.zip) available.

Posted Monday, November 25, 2002

CapeSoft Office Inside 1.0 Beta 1

After eight weeks and nine versions of alpha releases, beta 1 of CapeSoft Office Inside is now available.

Posted Friday, November 22, 2002

CPCS Previewer Addon Button Enhancements

CPCS has released new builds of v5.16 (C5) and v5.57h (C55) which have enhanced capability related to the newly implemented Previewer Addon Buttons. These buttons can now be associated with user-defined functionality (previously, they could only be used to control CPCS add-on products), and the visibility and functionality of each button can now be controlled at runtime. These builds are available to currently registered users of either v5.16 and/or v5.57h free of charge. Use the same install codes as the initial release.

Posted Friday, November 22, 2002

PdfWrite Class 2.0 Beta

The PdfWrite Class 2.0 beta release is now available. Use PDFWrite Class to generate Adobe PDF files on the fly. Everything is pure Clarion source without any black boxes. No OLE or API calls. You don't need to have Adobe or special printer drivers installed. Features include: multiple fonts (14 types); JPG images; text/background colors; automatic page numbering; different text regions on one page; various text effects (word/character spacing, scaling and rising). PdfWrite Class comes in two versions: Compiled DLL and Full Source. DLL version is $49.95, source version is $89.95. Upgrade price is $29.95 for the DLL version and $49.95 for the source version. Fully functional trial DLL version available.

Posted Friday, November 22, 2002

Clarion Companion Book On Sale

Randy Goodhew's book The Clarion Companion is currently on sale for half price through November 30, 2002. That's $37.50 plus shipping. Ground shipping in the U.S. is $4.50.

Posted Wednesday, November 20, 2002

INN Bio for 19-Nov-2002

This week IceTips presents a bio from an ex-engineer and ex-New Yorker who now spends his time enjoying warmer weather and programming in Clarion. This man has hobbies Sue Pichotta can identify with, and food shots our resident Cajun will applaud.

Posted Wednesday, November 20, 2002

EasyExcel 2.00 Demo

A new demo is available for EasyExcel 2.00. Changes in this release include: Fixed a bug with the incorrect cells addressing on some of the local Excel versions; Fixed a bug with SelectEnd method (in some cases it caused Excel to hang up); Chart methods allow you to create a charts in Excel; List2Excel exports the contents of the Clarion list box window control into Excel sheet; SetWritwDelay sets write delay on a fast PC (to prevent data loss during export); SetCellComment sets popup cell comment; WriteTXT writes multi-line text into the cell; SetSheetState hides/unhides specified sheet; Write code template now has cell comment and multi-line text export; List2Excel code template exports window list box contents to Excel; DrawChart code template creates a chart. Registered users of version 1.xx can get the upgrade for a discounted price at www.clarionshop.com.

Posted Tuesday, November 19, 2002

Gitano Software Holiday Schedule

Gitano's holiday schedule for the remainder of 2002 is as follows. Thanksgiving: Closed November 28 to December 1 - all sales and support from these days will be handled on December 2nd. Christmas: closed from December 24 to January 9. The Morenos will be in Spain during this time and support may be a little slow as email access will be limited.

Posted Tuesday, November 19, 2002

HTML Designer Help File Uploaded

Due to numerous requests, the help file for HTML Designer template set has been uploaded, and is available under the Purchase button on the Designer home page. This help file is in PDF format and describes the template set that comes with the HTML Designer WYSIWYG editor for creating HTML Help files for all versions of Clarion For Windows (32 bit).

Posted Tuesday, November 19, 2002

Icons XP Group II Released

Gitano Software has released the second in its series of icon packages. This new package contains 180 new images in ICO and GIF format. The icon resources contain several resolutions for optimal display in all color levels. Group II contains additional icons to match the ones in group one with the 'XP' look in two sizes and color schemes and a selection of smaller icons form lists and buttons. Price is $19.95. Save $5 when you buy Group I and Group II at the same time.

Posted Tuesday, November 19, 2002

PostgreSQL For Windows Beta

A beta of a native Windows version of the PostgreSQL server is now available. Created by PeerDirect, this port will reportedly be contributed to the official code base sometime in December, 2002.

Posted Friday, November 15, 2002

Free Utility Zips TPS Files

Chris Moore has a free utility to make it easier for customers to send you TPS files. The program asks the user to locate the folder that contains the TPS files. Once it is located, this utility will take all of the TPS files and put them in c:\Collected_Files.zip.

Posted Friday, November 15, 2002

Replicate Beta Released

The alpha test phase for Replicate is now over, and the beta has begun. Replicate provides an automatic, driver independent, file-version independent, mechanism for replicating the data in two or more databases. Basically, Replicate logs your changes, adds and deletes and then using a transport manager of your choice, exports the changes to the other sites, where the changes, adds and deletes are imported to that data set. This all done completely automatically without your users having to do anything. Replicate supports both offline and online environments. Features include: Track all adds/edits and deletes; Field-level replication only logs fields that have changed (when updating a record); Support for Logout/Rollback/Commit; Subset propagation - instead of replicating all the data to all children, Replicate can be configured to just send a subset of data pertaining to that child (or range of children); Variable synchronization timing - you can set up Replicate to synchronize as often as you would like, and you can do manual synchronizing with the push of a button; Complete data export; Table & field level suppression - suppress specific tables and/or fields that are not required (like calculated fields and temporary tables); Completely bi-directional replication; Optional Internal LogFile compression using Capesoft's zLib compression classes; Supports Legacy, ABC and hand-coded procedures and applications; Multiple version support - only imports those fields that are valid for the file version at the site (not driver dependant, so you can have a SQL file structure and a TPS file structure replicating to each other).

Posted Friday, November 15, 2002

New Dct2SQL Templates

Change in this release of the Dct2SQL templates include modifications by Jim Gambon to the Interbase template version.

Posted Friday, November 15, 2002

ImageEx2 Beta 1

ImageEx version 2.0 beta 1 is now available for download. Features include:; extended image control displaying images of more than 15 file formats (on windows AND reports!) with more than 39 extensions; fast and flicker free zooming and panning using the viewer control; a panoramic image viewer with user-defined, clickable hotspots; a wrapper class for creating, loading, modifying & saving 32bit images (24bit RGB plus 8 bit alpha channel); image processing functions like adjusting contrast, brightness and intensity, gamma adjustment & color correction; resizing, rotating, flipping and skewing images; special effects like mosaic

filter, convolution filters and colorizations; royalty free TWAIN scanning without any third party files; advanced drawing functions (line, rect, roundrect, ellipse, polygon, pie, etc.) with alpha channel support and antialiasing for smooth results; utility functions like the PictureDialog (a file dialog with a preview area for images), others for saving images to and loading from BLOBs (without the need for an image control), retrieving properties from image files. Demo available. The new version is available at ClarionShop, http://www.clarionshop.com, for $149 US (beta price, the gold version will sell at $199). Existing 1.x users can upgrade for $49 during beta, $99 afterwards (you won't have to buy the update if you don't need the advanced imaging functions). There will be a "lite" version of ImageEx 2.0 soon that will be free of charge for 1.x users.

Posted Friday, November 15, 2002

Save On gFileFind

For a limited time, you can save $30 on Gitano's gFileFind, regular $79.95, now $49.95. Included with your purchase is a free PowerSearch license.

Posted Tuesday, November 12, 2002

ABXTrackbar Released

ABX Systems has released ABXTrackbar, a class wrapper and templates for the trackbar common control. Price is $89.

Posted Tuesday, November 12, 2002

RInstall HTML Designer feedback.

The Rinstall templates have been updated. Customers can download using the same password as issued by ClarionShop. Some issues regarding multi users off a single application running on a server have been addressed. The previous release also unfortunately had a few read-only files attached. this has been fixed.

Posted Tuesday, November 12, 2002

xFText v1.02 Released

SealSoft has released xFText v1.02. New features include: Legacy template support; Modification in main extension template to add label for text; New methods to add, change and remove frame text in runtime; New code templates: ChangeText, RemoveText, RedrawText. New demo and install available.

Posted Tuesday, November 12, 2002

Clarion Handy Tools Email Demo

The new Clarion Handy Tools Build (O7B2) demo includes the ability to easily send SMTP mail and compress/encrypt attachments. Compression/encryption (up to 448 bit) works with the CHT MAPI email implementation as well.

Posted Tuesday, November 12, 2002

Steven Muller's Email Merge Example On Par2

Steve Parker has placed Steven Muller's eMail/RTF merge sample application on the Par2 download site.

Posted Tuesday, November 12, 2002

EMS PostgreSQL Manager

EMS has released PostgreSQL Manager, which has the same look and feel as the company's popular MySQL Manager SQL client.

Posted Tuesday, November 12, 2002

SMTP Demo

Ville Vahtera has a demo of an SMTP agent available for download.

Posted Tuesday, November 12, 2002

Service Pack #1 for TX Text Control v10

The Imaging Source Europe released an intermediate service pack that fixes several issues. SP1 fixed printing issue under 95/98/98SE/ME which was also affecting streaming document to Windows metafiles. More info about Clarion wrapper for the TX Text Control and updated Clarion examples are available at the web site. Trial version available.

Posted Tuesday, November 12, 2002

TemplateSpy Released

IntelliNova has released TemplateSpy, a programmer's utility created to assist in creating and managing templates It's features include the following: View the template structure of registered templates in an Explorer pane, with the text of the selected element in a Source pane; Facilitate the editing and creating of templates via drill-through to your favorite text editor, and copy and paste; Assemble full template source - across multiple template files; Search through a template or template component for a text string; search through multiple source files for a text string; Compare template (and Clarion) source version differences; Identify which applications are using which templates, and in which procedures. Available at ClarionShop for $99.

Posted Tuesday, November 12, 2002

Email Merge Freeware

Steven Muller has put together a simple application that uses the standard RTF and SMTP controls to produce merged email letters. This application is derived from a course project he has his students working on at the moment. A toolbox allows you to paste merge fields into the RTF document. These merge fields are then replaced with appropriate recipient data when the email messages are produced. The email letters are then sent as plain text. There are no

instructions or documentation. Email Steven for a copy.

Posted Tuesday, November 12, 2002

BlinkFlash Freeware

BlinkFlash, now released as freeware, is a an extension template which puts a "blinking attribute" on a control and allows an unlimited number of controls per window to blink. Options include: Delay before blink starts; How long to blink; Whether text or background should blink; Frequency of blink; Conditional blinking - such as "SALES < 10"; Multiple controls per window. Compatible with CW2002 to C5.5, ABC and Legacy. Demo apps supplied.

Posted Tuesday, November 12, 2002

INN Bio & News For 05-Nov-2002

This week, the Icetips News Network is pleased to present a first bio from Russia! A Clarionite since forever, this developer is now known for his involvement in third-party tools. When he's not using Clarion, he's spending time with his family and taking photos, many of which you can see in this bio.

Posted Tuesday, November 05, 2002

SetupBuilder 4.02b Web Edition (Beta 1.5)

The Beta 1.5 release of SetupBuilder 4.02b Web Edition is available now. The Web-Install technology allows you to very easily deploy (and in V5.0 update) your applications over the Internet. The installation runs as a stand-alone application and uses the WINSOCK API to download the necessary files required for the installation. SetupBuilder does not rely on browser plug-ins or ActiveX controls. Beta 1.5 fixes some minor bugs. This beta will be the last 4.x release. The beta release of SetupBuilder 5.0 is nearly feature complete and the Web capability has already been implemented in SetupBuilder 5.0 Professional Edition. To accommodate the increased development efforts the price of SetupBuilder 5 will be going up. Download SetupBuilder 4.02 now and buy it before the price goes up - the upgrade to SetupBuilder 5.0 will be free of charge.

Posted Tuesday, November 05, 2002

CPCS v5.16 And v5.57h Now Available

This upgrade for Clarion C5 (v5.16) and C55 (v5.57h) contains a number of often asked for enhancements to the Creative Reporting Tools product. New purchase or upgrade entitles you to both v5.16 (C5) and v5.57h (C55). Current users who purchased or upgraded to either v5.15 or v5.50d after Sept 30, 2002, are entitled to this upgrade free. Please contact support@cpcs-inc.com. New features include: Access the Progress Window via the window formatter; Pause Control template added for use on the Progress Window; Up to 3 CPCS add-ons can be controlled via buttons on the Preview; WinWord style scrolling option added to Previewer; String Searching added to Previewer; Different sized pages will now display properly in the

Previewer; Global "Nothing to Preview/Print" option added to reports; Help file now opens properly under Windows XP.

Posted Tuesday, November 05, 2002

SCA Micro Templates

Changes in this release include: Third release of the Browse Header Sort template, now supports single click on header; New template which navigates automatically from the last field of a tab to the first field of the next tab; New template with native XP look for checkboxes, option boxes and buttons without icons; New demo.

Posted Tuesday, November 05, 2002

ThinkData Releases qbFUSE QuickBooks COM Automation

ThinkData, Inc. has released qbFUSE, a native COM (Component Object Model) automation interface to QuickBooks using Clarion 5.5. qbFUSE integrates with QuickBooks using the QBFC (QuickBooks Foundation Classes) SDK, and relies on the professional version of the Plugware COM classes. qbFUSE features include: Native early and late binding COM automation; Professional Version of the Plugware COM Classes; Full access to over 400 interfaces provided by the QBFC SDK; Full source code to examples included; Ability to integrate QuickBooks using Clarion with the exact same functionality that can be achieved using Visual Basic or Visual C++; Free technical support via online forum. Cost is US$199.

Posted Tuesday, November 05, 2002

Image-XChange SDK Released

Tracker Software has released the first fully functional version of the Image-XChange SDK for Clarion developers! This is fully functional but will print a watermark until the developer key has been purchased.

Posted Tuesday, November 05, 2002

Taboga Software Holiday Schedule

Edgard Riba will be on holiday from November 1-13. He'll try to read mail at taboga@rimith.com, but this will be sporadic as he'll be out of the country.

Posted Tuesday, November 05, 2002

ImageEx 2 Preview Demo

A preview demo of the upcoming ImageEx 2.0 is now available. The main feature of this second version is ImageExBitmapClass, a Clarion representation of a 32bit (24bit RGB color, 8 bit alpha channel) device independent bitmap with lots of functions to load, modify and save images: ; Loads from more than 16 file types including TIF, PNG, PCD; Saves to BMP, JPG, TGA (more to come); Acquires from TWAIN devices (no extra DLLs or ActiveX required, royalty free); Adjusts brightness, contrast & intensity; Adjusts gamma as well as RGB colors; Supports 3x3 Convolution filters; Does graphic primitives like rect, round rect, ellipse, pie, arc,

chord, line, polygon with alpha channel & antialiasing(!); has advanced bitmap functions like affine & projective transformations; lets you access each pixel individually. Other new features include a PaintBox control and a PictureDialog. The price of 2.0 will be US$199; updates from version 1.x will be US$49.

Posted Tuesday, November 05, 2002

Reader Comments

Add a comment

Determining Gender With Clarion (Part 2)

by Geoff Robinson

Published 2002-11-19

Last week I discussed the events that led me to write a gender guessing function in Clarion, and covered regular expressions and a first attempt at that function. Now it's time to refine and test the code.

After I'd written my initial version of the function, I decided to put the regexes into the data area. This also allowed me to only do the match if the result could possibly change the current state of the sex. For example, if the current state of variable s (for sex) is 'f' (for female) then there is no point doing a regex if it results in 'f' anyway….

Note that in the data below all the right hand quotes and comments line up nicely. If I had stuck to curly braces this would not have been the case and errors could easily creep in. Again there is a certain elegance about it. I defined the data as one long string (broken visually up into the 40 regexes) and then defined the table over the string.

```
VitGetGender2        FUNCTION (STRING p:Name)
s   STRING('m')   ! sex defaults to male...
REData STRING( |
 '^.*[aeiy]$                          f' & | most names ending in a/e/i/y are female
 '^All?[iy]((ss?)|z)on$               f' & | Allison and variations
 '^.*een$                             f' & | Cathleen, Eileen, Maureen
 '^[^S].*r[rv]e?y?$                   m' & | Barry, Larry, Perry
 '^[^G].*v[ei]$                       m' & | Clive, Dave, Steve
 '^[^BD].*(b[iy]|y|via)nn?$           f' & | Carolyn, Gwendolyn, Vivian
 '^[^AJKLMNP][^o][^eit]*([glrsw]ey|lie)$ m' & | Dewey, Stanley, Wesley
 '^[^GKSW].*(th|lv)(e[rt])?$          f' & | Heather, Ruth, Velvet
 '^[CGJWZ][^o][^dnt]*y$               m' & | Gregory, Jeremy, Zachary
 '^.*[Rlr][abo]y$                     m' & | Leroy, Murray, Roy
 '^[AEHJL].*il.*$                     f' & | Abigail, Jill, Lillian
 '^.*[Jj](o|o?[ae]a?n.*)$             f' & | Janet, Jennifer, Joan
 '^.*[GRguw][ae]y?ne$                 m' & | Duane, Eugene, Rene
 '^[FLM].*ur(.*[^eotuy])?$            f' & | Fleur, Lauren, Muriel
 '^[CLMQTV].*[^dl][in]c.*[ey]$        m' & | Lance, Quincy, Vince
 '^M[aei]r[^tv].*([^cklnos]|([^o]n))$ f' & | Margaret, Marylou, Miri;
 '^.*[ay][dl]e$                       m' & | Clyde, Kyle, Pascale
```

```
  '^[^o]*ke$                              m' & | Blake, Luke, Mi;
  '^[CKS]h?(ar[^lst]|ry).+$               f' & | Carol, Karen, Shar;
  '^[PR]e?a([^dfju]|qu)*[lm]$             f' & | Pam, Pearl, Rachel
  '^.*[Aa]nn.*$                           f' & | Annacarol, Leann, Ruthann
  '^.*[^cio]ag?h$                         f' & | Deborah, Leah, Sarah
  '^[^EK].*[grsz]h?an(ces)?$              f' & | Frances, Megan, Susan
  '^[^P]*([Hh]e|[Ee][lt])[^s]*[ey].*[^t]$ f' & | Ethel, Helen, Gretchen
  '^[^EL].*o(rg?|sh?)?(e|ua)$             m' & | George, Joshua, Theodore
  '^[DP][eo]?[lr].*s$                     f' & | Delores, Doris, Precious
  '^[^JPSWZ].*[denor]n.*y$                m' & | Anthony, Henry, Rodney
  '^K[^v]*i.*[mns]$                       f' & | Karin, Kim, Kristin
  '^Br[aou][cd].*[ey]$                    m' & | Bradley, Brady, Bruce
  '^[ACGK].*[deinx][^aor]s$               f' & | Agnes, Alexis, Glynis
  '^[ILW][aeg][^ir]*e$                    m' & | Ignace, Lee, Wallace
  '^[^AGW][iu][gl].*[drt]$                f' & | Juliet, Mildred, Millicent
  '^[ABEIUY][euz]?[blr][aeiy]$            m' & | Ari, Bela, Ira
  '^[EGILP][^eu]*i[ds]$                   f' & | Iris, Lois, Phyllis
  '^[ART][^r]*[dhn]e?y$                   m' & | Randy, Timothy, Tony
  '^[BHL].*i.*[rtxz]$                     f' & | Beatriz, Bridget, Harriet
  '^.*oi?[mn]e$                           m' & | Antoine, Jerome, Tyrone
  '^D.*[mnw].*[iy]$                       m' & | Danny, Demetri, Dondi
  '^[^BG](e[rst]|ha)[^il]*e$              m' & | Pete, Serge, Shane
  '^[ADFGIM][^r]*([bg]e[lr]|il|wn)$       f')  ! Angel, Gail, Isabel

REGroup GROUP,DIM(40),OVER(REData)
Exprn   STRING(39)
Sex     STRING(1)
   END !group

i  LONG

  CODE
    if ~p:Name then return('?').  ! no name passed...
    p:Name = lower(p:Name)
    p:Name[1] = upper(p:Name[1])  ! capitalise name
    LOOP i = 1 TO MAXIMUM(REGroup,1)
      ! only do the RE if the result can change current value of sex
        if s <> REGroup.Sex[i]
            if VitRegMatch(p:name,REGroup.Exprn[i])
                s = REGroup.Sex[i]
!               vitdebug(clip(p:name) & ' sex changed to ' |
                  & s & ' via rule ' & clip(REGroup.Exprn[i]))
            end
        end
    END !loop
    return(s)
```

The commented out line writes a debugging trace to a file; this is handy if you're trying to follow which rules generated a result. `Vitdebug()` is just my logging function, so substitute your own here.

Note the use of `MAXIMUM()`. It is always good to use this on arrays, just as you should use `SIZE()` when looping a character at a time through a string. `MAXIMUM()` aids maintainability and means you are less likely to introduce bugs when your table changes size.

This version worked, but I was conscious of the fact that I was checking for those parentheses each time I called `VitRegMatch`. It would be better to do it just once. The next version made `REData STATIC` and added a single byte `REConverted` to the data area:

```
...
REGroup GROUP,DIM(40),OVER(REData)
Exprn   STRING(39)
```

```
Sex     STRING(1)
   END !group
i  LONG
REConverted  BYTE,STATIC
  CODE
    if ~p:Name then return('?').  ! no name passed...
    if ~REConverted
        ! replace parentheses () with curly braces {}
        LOOP i = 1 TO SIZE(REData)
            CASE REData[i]
              OF '(' ; REData[i] = '{{'
              OF ')' ; REData[i] = '}'
            END !case
        END !loop
        REConverted = TRUE
    end
    p:Name = lower(p:Name)
    p:Name[1] = upper(p:Name[1])  ! capitalise name
    LOOP i = 1 TO MAXIMUM(REGroup,1)
         ! only do the RE if the result can change
                 ! current value of sex
         if s <> REGroup.Sex[i]
            if Match(p:name,REGroup.Exprn[i],Match:Regular)
                s = REGroup.Sex[i]
!               vitdebug(clip(p:name) & ' sex changed to ' |
                  & s & ' via rule ' & clip(REGroup.Exprn[i]))
            end
        end
    END !loop
    return(s)
```

The trick here is the use of `STATIC`, which means the data survives between invocations. In other words when you call the function it still has the value preserved in it from the last time it was called. Hence I can scan the regexes and do the conversions just once. Just to be different, this time I have manually looped and used a `CASE` statement to check for parentheses – after which `REConverted` is set to `TRUE`. Having done this I no longer need to call `VitRegMatch()` and can call `Match()` directly.

Accuracy

How accurate is the function? Firstly I should point out that any gender guessing algorithm can never be perfect. Some names like Jessie, Frankie, Kerry, Leslie, Lindsay, Marion and Rickie are transgender – both males and females use them. To complicate things further, sometimes this is the case with shortened names too. For example I know a Beverly (female) and a Bevan (male) both of whom are referred to as "Bev". Same goes for Alex, while other names like Jan could be short for Janette (female) or pronounced *Yarn* or *Yan* and be male. Jean could be female or could be Monsieur Jean. You get the general idea.

I grabbed some test data and ran some tests. Initial tests against a database of about 120,000

people where I already had a title field (eg. Mr., Mrs., Miss., Ms. etc.) gave an accuracy rate of about 92%. Not bad, I thought, but room for improvement.

At this stage I had to decide whether to fiddle with the regexes, altering some and adding new rules here and there, or introduce exceptions. I decided to take the easy way out and provide a list of exceptions.

If you look at the source code for `VitGetGender4` you will see I've added a big case statement, of the form:

```
CASE p:Name
OF   'Abi' OROF 'Abigayle' OROF 'Aerial' |
OROF 'Ambar' OROF 'Anais' OROF 'Ashleigh'
  ...
OROF 'Yanet' OROF 'Yasmin' OROF 'Yazmin' |
OROF 'Yoko' OROF 'Zoe'
  s = 'f'
OF   'Abel' OROF 'Adonis' OROF 'Alejandro' |
OROF 'Alijah' OROF 'Andre' OROF 'Anfernee'
  ...
OROF 'Yovani' OROF 'Zaire' OROF 'Zane'
  s = 'm'
ELSE
  LOOP i = 1 TO MAXIMUM(REGroup,1)
    ! only do the RE if the result can
    ! change current value of sex
    if s <> REGroup.Sex[i]
      if Match(p:name,REGroup.Exprn[i],Match:Regular)
        s = REGroup.Sex[i]
        ! vitdebug(clip(p:name) & ' sex changed to ' & s |

        ! & ' via rule ' & clip(REGroup.Exprn[i]))
      end
    end
  END !loop
END !case
```

I automatically generated much of the code for the exceptions by writing it out based on my test data, and also by reviewing some lists of names from the Internet. I then added some more exceptions by hand as I came across them. When checking my data I found that many of the name entries that were generating errors were in fact spurious. Some appeared to be just data entry errors, whereas some came about because of names like *Mrs. John Smith.* (I must say that in this day and age it seems a fairly archaic practice where a wife is officially known by her husband's given name *as well as* his surname - but ours is not to reason why…)

Actually talking of names lists, if you look at the US Census Bureau site you can get name lists of female, male and surnames showing their frequency distribution. Did you know, for example, that *James* is the most common male name (in the US)? (I would have guessed *John*

which actually comes in second….) And *Mary* is the most common female name by a large margin (outnumbering the combined total of second and third placeholders *Patricia* and *Linda*).

With my big list of exceptions coded, I re-ran the tests and was initially surprised that the accuracy rate had only increased a little over one percent to just over 93%. On reflection this is because the names in the exception list are, in the main, not that common in my data.

At this stage I was wondering about the efficiency of having such a large Case statement and thinking that perhaps I would be better to load the exceptions into a static queue sorted on name that also stored sex. I could do that at the same time that I replaced the parentheses with curly braces, and that way I could simply do a `GET()` on the name in the queue to see if it was an exception. This is a classic case where there would be a higher up-front performance cost that you hope to make up as the program runs. It all depends how often you are going to call the function.

Another alternative would be to put the names and corresponding sex into an array. As with the regexes, I could define a string field initialised with the data and then declare the array `OVER` the string. Assuming I entered the names in alphabetic order I could then do a binary search of the array. Of course writing more code complicates things and can also lead to bugs. Remember that *the code with the least bugs is the code you do not write!*

Another option would be to use `instring()` as in :

```
if instring(clip(p:name),|
  'AbiAbigayleAerialAmbarAnaisAshleigh…etc',1,1)
    s = 'f'
elsif instring(clip(p:name),|
  'AbelAdonisAlejandroAlijahAndreAnfernee..etc',1,1)
    s = 'm'
else
    ! do RE code
end
```

Note that I do not need to separate the names with a space or comma etc. due to the fact that I have capitalised them. While I haven't tried this `instring()` version, I suspect it might be slower than the `Case` statement as, assuming a brute force implementation, `instring()` needs to check every character position rather than just from the start of each name. Mind you I've always found `instring()` to be quite fast… TMTOWTDI

For the moment I've decided to leave the `Case` statement.

At the risk of harping on about associative arrays, if Clarion had them I would be able to simply say something like:

```
Sex['Abi'] = 'f'
Sex['Abigale'] = 'f'
Sex['Abel'] = 'm'
Sex['Adonis'] = 'm'
```

...and so on for the other exceptions. In fact in Perl (where an associative array is usually referred to as a *hash*), you can put many key/value pairs in the one assignment. A Clarion equivalent might be along the lines of:

```
Sex = [‘Abi’, ‘f’, ‘Abigayle’, ‘f’, ‘Abel’, ‘m’, ‘Adonis’, ‘m’]
```

Or borrowing Perl’s alternative separator => instead of a comma (for readability):

```
Sex = ['Abi'       => 'f',
       'Abigayle'  => 'f',
       'Abel'      => 'm',
       'Adonis'    => 'm']
```

Then in my code I could simply say:

```
s = Sex[p:Name]
if ~s
!  do the  RE code as no exception found
end
```

If wishes were fishes…

Conclusion

Gender-guessing is a complex task and one to which, on the surface, Clarion would not seem particularly well adapted. The code I used as a basis was originally written in AWK and later translated to Perl. All languages have their strengths and weaknesses, and both these languages have associative arrays which would be a welcome addition to Clarion. The gender guessing made great use of regular expressions and I was able to adapt this code to Clarion by using the MATCH function’s regular expression capability and then improve it further by moving the rules to the data area, where I implemented them as arrays initialized by declaring them OVER predefined strings.

Clarion comes with a large number of powerful functions built in, and most programmers, over time, create custom functions which they use again and again. And now you have a new gender guessing function to add to your own code library.

Download the source

Geoff Robinson lives near the beach in Melbourne, Australia, and is an active member of the local Clarion User Group. His company, Vitesse Information Systems, specializes in software for local government. Geoff was impressed by Clarion back in the DOS days and grabbed the early betas of Clarion for Windows when they first became available; he has been using Clarion as his primary development environment ever since. When not in front of a computer Geoff enjoys listening to music, singing bass in a local choir, and spending time with his three young children.

Reader Comments

Add a comment

Staying Connected On The Road

by Ned Reiter

Published 2002-11-21

If you're like a lot of Clarion software developers, you spend at least some time on the road, and, you no doubt have your favorite way of staying connected to the online world. If you stay in hotels, or with friends or family, a phone line is never far away. But what if you travel by recreational vehicle (RV)? In 1997, my wife and I bought a 38' motor home, and we've been on the road full time ever since. I've learned a lot in that time about getting online; in this article, I'll explore the connection options available to RVers.

Five years ago, about the only option for getting to the Internet while RVing was to borrow a phone line. Some campgrounds had jacks for modem connections, but most did not. Truck stops were another option, but that required taking your computer into the restaurant, and sometimes the phones were fastened down and you weren't able to get to the RJ-11 jack. Over the next few years, more campgrounds added phone jacks, usually in the office, game room or laundry, just for modem use, and some even added phone jacks to some or all of their camp sites. These days a typical charge for a "hot" site is $1-2 per day for the phone line. You are restricted to local or 800 calls only, of course. Right now, there are about 450 campgrounds in the continental US that have hot sites and several thousand that have an available phone jack for computer use. These campgrounds are referred to as "modem friendly."

For RVers who stay in one location for a month or more, many campgrounds have the camp sites wired for telephone service – you only need to contact the local telephone company to order a phone line. This can get expensive if you move frequently, as installation charges run about $50 every time you order. Don't think you can get DSL or cable modem service in a campground. Most DSL and cable providers will require a one year contract, and many campgrounds are too far away from the nearest phone company switch to support DSL. That eliminates broadband as an option for the frequent traveler.

Another option is a cell phone/cellular modem combination. A few years ago, a popular combination was a Nokia 51xx or 61xx series phone with the 3Com 3CXM556 modem and the AT&T Wireless Digital One Rate cellular plan. This plan allowed you to make calls anywhere in the U.S. that you could get an analog cellular signal, with no roaming or long distance charges. The data rates were limited to 4800bps for reliability, or up to 9600bps with more errors, and only on an analog connection. Many RVers are still using this method today. Unfortunately, it's getting very difficult to get the needed cable for this system as the cellular providers are all pushing the digital data services. In fact, none of the phones currently available from AT&T Wireless, Sprint or Verizon Wireless are capable of analog data.

With the advent of the digital data services, connectivity has improved. As an example, the Verizon America's Choice cellular plan with the right phone and cable lets you connect at 14400bps whenever you are in a Verizon digital or digital partner area. Web browsing is still painfully slow, but it is can be done. The newer, faster, digital services improve the speeds to those of dialup or faster, but are expensive and are only available in some areas, mostly the larger cities.

The cell phone solutions are adequate for email and small file transfers, but a developer also needs access to the web and newsgroups, and often needs to both upload and download large files. A faster solution is now at hand.

Satellite internet access has been available for some time, but it required a professional installation at a fixed location by a trained installer. Some services were one way only, and required a telephone line for the uplink. None of these services is suitable for a person who may move as often as every day, and lives in campgrounds.

Last year, Motosat announced they were developing a mobile internet access product, the DataStorm. This is a true two way satellite internet system, using DirecWay from Hughes Network Systems. Motosat's contribution is an autolocating mount for the satellite dish. The system was in beta test for quite a few months, and was released to dealers earlier this year. It does require professional installation and isn't cheap. The hardware has a MSRP of $6995 but is available from dealers for under $6000. Installation will cost about $1000.

The Motosat/DirecWay system is quite a change from cell phones and land lines. The download speeds vary from 100kbps to over 1Mbps, and upload speeds range from 40-90kbps. As with any broadband connection, it's always on. Operation is easier than hooking up a modem. After parking your RV and leveling it, you run a program on your PC and click on the **Find Satellite** button. About 7-10 minutes later, you're connected to the internet. The system uses NAT (Network Address Translation) so your PC has a non-routable address, making you invisible to anyone outside of your local network. It's just like being behind a firewall. Internet Connection Sharing works as well, so you can share the connection with other computers on

your LAN. Sitting on the patio, surfing the web at broadband speeds without any wires (via WiFi) really impresses people!

In addition to the hardware and installation costs, the service is $99/month or more, depending on the level of service needed and if you want a static IP address. For most rvers, the minimal Business Edition Basic at $99/month is the right choice.

I have used the Motosat/DirecWay system for several weeks now and I can't imagine ever going back to a dialup connection, let alone the cellular phones. We still have two different cellular phones and service plans, but they're only used for voice calls now. With the satellite dish system, I don't even need to be in a campground. I can run the system from my batteries and inverter while parked in the remotest desert location.

As you can see, the choices are varied in both usability and cost. If all you need is email, then the cellular connection is more than adequate. But if you need a full time internet connection and the ability to transfer large quantities of data, then the Motosat/DirecWay satellite system is the only solution available today that will work wherever you may be parked.

Ned Reiter has been programming since 1961 and a Clarion user since CPD 2.0. After spending the first 53 years of his life in the Milwaukee, WI, area, he and his new bride, Lorna, moved to Palm Springs, CA, in 1995. Deciding that life in CA wasn't to their liking, the sold nearly everything and bought a 38' motor home in 1997 and went on the road full time. They are still enjoying life on the road while Ned works from the motor home, writing mostly custom Windows applications in Clarion for Windows.

Reader Comments

Add a comment

Currently I have a the Sprint Vision Service with a...
Mike, Vision is a good option IF you spend most of all...

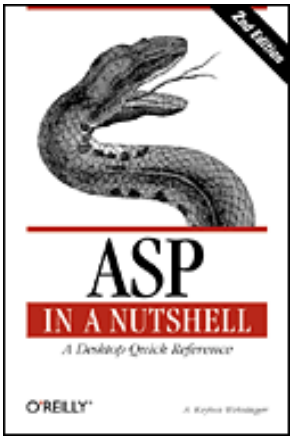
Clarion Magazine

Reborn Free CLARION online

Books Reviewed: Getting Your ASP In Gear

by David Harms

Published 2002-11-22



ASP in a Nutshell, 2nd Edition - A Desktop Quick Reference

By A. Keyton Weissinger
Published by O'Reilly, 2nd Edition July 2000
ISBN: 1-56592-843-1
492 pages, $29.95 US, $43.95 CA, £19.95 UK

Designing Active Server Pages

By Scott Mitchell
Published by O'Reilly, September 2000
ISBN: 0-596-00044-8
360 pages, $29.95 US, $43.95 CA, £20.95 UK

Developing ASP applications is a *lot* different from developing Clarion desktop applications. And while Clarion/ASP has done much to bridge the gap for Clarion developers, it's still a whole new world out there for most of us. This week I'll take a look at a couple of books from O'Reilly that aim to bring intermediate developers up to speed on ASP.

Designing Active Server Pages is written by Scott Mitchell, one of the founders of the popular

ASP site 4GuysFromRolla.com. Mitchell describes the book as intended for "intermediate to advanced Active Server Pages developers who have solid ASP skills," but I think any relatively experienced Clarion developer with some knowledge of HTML would be able to grasp the book's concepts.

Mitchell begins by asking (rhetorically) what is application design, what's wrong with ASP design, and what can be done to improve ASP design? The problem, he believes, is poor planning and lack of code reuse. It's easy enough to write an ASP page by slapping a bunch of VBScript code into an HTML page and changing the extension from .html to .asp. But that leaves the code mixed in with the presentation (HTML). Mitchell's main push is to keep the scripting code in server side includes, although he also sees VBScript classes (available as of version 5.0) and branching control to other ASP pages as important techniques.

Chapter 2 covers scripting languages. VBScript is the most popular choice for ASP programming, and virtually all of the examples in this book use that language. Chapter 3 discusses error/exception handling, again, mainly in the context of VBScript, and Chapter 4 looks in detail at several aspects of VBScript programming, including regular expressions, classes, and dynamic expression evaluation and execution. That wraps up the first third of the book. You won't find an extensive VBScript tutorial here, but then again if you can read Clarion source code and you're familiar with HTML, VBScript shouldn't pose too many problems.

The middle third of the book is taken up with a chapter on forms and another lengthy chapter on databases. By the time you've digested these two chapters, you should know how to build a set of web pages to administer a variety of database tables. And there's the rub – if you're a Clarion/ASP user, you'll generate all that functionality with Clarion. There are pages and pages of VBScript source code that you'll never need. On the other hand, you may pick up a few tricks from these chapters that you can use in your own embedded ASP source; certainly the background on ASP form handling is useful information, in particular the discussion of client side versus server side validation (to which all web developers should pay special attention).

The last third of the book looks at component re-use. There is one chapter devoted to COM in general, and the standard Microsoft ASP components in specific, including the Ad Rotator, Content Linker, Content Rotator, Browser Capabilities, Counters, Permission Checker, MyInfo server information, and the Tools collection. The final chapter explores a variety of third party components, including: ASPExec, which lets an ASP page execute and application on the web server; cyScape's BrowserHawk, which offers detailed information on users' browser capabilities; and Stephen Genusa's ASPHTTP object, which pulls information from other HTTP servers. Other topics covered are encryption and digital certificates, and uploading files using multi-part forms.

The bottom line

Designing Active Server Pages is readable and presents information clearly, with lots of links for further study. It isn't a rank beginner's book, and unfortunately for Clarion developers two of the meatiest chapters are rendered largely irrelevant by Clarion/ASP, but it's still worth a look.

* * *

The other ASP book in this review is O'Reilly's *ASP In A Nutshell*, subtitled A Desktop Quick Reference. I find the title a bit misleading – you might think that a subject in a nutshell would mean an overall, high level view. That really isn't the case. This book is what the subtitle says: it's a reference book, spiced with lots of good advice on specific situations.

The first section of the book, which comprises three chapters, *is* titled Introduction to Active Server Pages, and while the information presented in these chapters is accurate, this is the one part of the book which I suspect readers may find the least helpful. If you already have some HTML and Visual Basic or VBScript experience, and you have a rough idea of how web applications work, then these chapters will fill some gaps in your knowledge, but they won't give you the larger picture of how to put together an ASP application. Instead, the book goes straight from the obligatory intro into the reference section. And that's probably just as well; if you want a step by step tutorial on ASP programming, you should be looking elsewhere.

Part II covers the each of the core ASP objects, in turn. These include the Application, ASPError, ObjectContext, Request, Response, Server, and Session objects. From here on in, each chapter beings with a Comments/Troubleshooting section. For instance, the chapter on the Request object begins with an explanation of the HTTP protocol that web servers and clients use to communicate. The chapter on the Response object gives a short but highly useful tip on debugging generated web pages. Help files will give you the nuts and bolts of the various components; what you're paying for in this book are the many tips and explanations mixed in with the reference material.

Part III details the Microsoft installable components, including those discussed by *Designing*, and with the addition of ADO, Collaboration Data Objects (CDONTS), and the Logging Utility. At some 85 pages, the chapter on ADO is by far the longest in the book, and with good reason. ADO is a complex topic, and as it is the way Clarion/ASP applications talk to databases it's worth some study, especially if you plan to write any embedded data access code. The CDONTS component, which *Designing* briefly mentions, is also worth a look; it uses SMTP or Microsoft Exchange to send email, and can do so with a minimum of fuss and bother. The Logging Utility component lets you access, and generate reports on, the IIS or FTP server logs.

The bottom line

ASP In A Nutshell probably won't be the first ASP book you'll pick up, but if you do any amount of ASP coding, you'll want to have a copy handy. It's easiest to read if for each chapter you start with the introductory paragraphs, followed by the Comments/Troubleshooting section, and then the rest if/as needed.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995). His most recent book is JSP, Servlets, and MySQL, published by HungryMinds Inc. (2001).

Reader Comments

Add a comment

Introducing PostgreSQL (Part 1)

by David Harms

Published 2002-11-28

In the world of open source databases, there are two long-standing rivals, MySQL and PostgreSQL, and one new contender, Firebird, which is the open source version of Interbase. I've written a number of articles about MySQL, and I may at some future time write about Firebird. In this new series of articles, however, I'll be talking about PostgreSQL, and providing a diary of sorts of my attempt to migrate one database from MySQL to PostgreSQL.

PostgreSQL has its roots firmly in the University of California at Berkley. Its ancestor is Postgres, an object-relational database developed at the university in the late 80s and early 90s. Postgres used a query language called POSTQUEL; in 1994-5, Jolly Chen and Andrew Yu, graduate students at Berkley, added SQL capabilities and called the database Postgres95. In 1996, Marc Fournier volunteered to host the server for the source tree and the mailing list, and Postgres95 became PostgreSQL. Since that time, the PostgreSQL open source community has clearly flourished, and PostgreSQL has become quite popular.

But not as popular as MySQL, it would seem. MySQL has, for some time, been a darling of the computer press, an up-and-comer threatening the big database vendors. And this has gotten up the noses of some PostgreSQL supporters. Why, they ask, does everyone talk about MySQL, when PostgreSQL has more "real" SQL database features, like views, sub-selects, transactions (okay, MySQL has those now) and so forth?

Part of the answer, I think, is that MySQL has, for years, had a native Windows version, in addition to versions that run on the many Unix/Linux platforms. With PostgreSQL, you could run on just about any hardware, but if you wanted to run on Windows, you had to do so inside the Cygwin Unix environment for Windows. Ugh. Can you say "emulation?"

I'm happy to report that a native Windows version of PostgreSQL has now arrived. Well, at least you can get a beta at ftp://209.61.187.152/postgres/postgres_beta4.zip. This version was developed by PeerDirect, and will reportedly be contributed to the PostgreSQL project in December 2002. Interestingly, the beta is on a NuSphere server, and NuSphere (a subsidiary of Progress Software, as is PeerDirect) recently settled a lawsuit launched by MySQL AB involving NuSphere's creation of a www.mysql.org web site, its use of trademarks, and its alleged failure to release the Gemini database handler under the GPL.

Open source database internecine warfare aside, the PostgreSQL folks say on their advocacy site that the Windows version will be part of the official distribution as of version 7.4. The current release is 7.2.3, and 7.3 is in beta.

Installing the PostgreSQL Windows beta

The beta is quite easy to install – just unzip everything into a directory, and then modify the setenv.bat file (which you'll find in that directory) accordingly. My setenv.bat looks like this:

```
set PGHOME=d:\postgres_beta4
set PGDATA=%PGHOME%\data
set PGLIB=%PGHOME%\lib
set PGHOST=localhost
set PATH=%PGHOME%\bin;%PATH%
```

Run the batch file from a command prompt, not from Windows, because after you set the environment variables you'll need to initialize the database by running the initdb utility. Here's the output on my machine:

```
D:\postgres_beta4>initdb
The files belonging to this database system will be owned by user "dharms".
This user must also own the server process.

creating directory d:\postgres_beta4\data...ok
creating directory d:\postgres_beta4\data\base...ok
creating directory d:\postgres_beta4\data\global...ok
creating directory d:\postgres_beta4\data\pg_xlog...ok
creating directory d:\postgres_beta4\data\pg_clog...ok
creating template1 database in d:\postgres_beta4\data\base\1
 ...creating configuration files...ok
initializing pg_shadow...ok
enabling unlimited row size for system tables...ok
creating system views...ok
loading pg_description...ok
Installing PeerDirect UltraSQL Replication Adapter Support
vacuuming database template1...ok
copying template1 to template0...ok

Success. I could now start the database server using:

d:\postgres_beta4\bin\postmaster -D d:/postgres_beta4/data
```

or

```
d:\postgres_beta4\bin\pg_ctl -D d:/postgres_beta4/data
    -l logfile start
```

I put the start command into start.bat and gave it a whirl.

```
d:\postgres_beta4\bin\postmaster -D d:/postgres_beta4/data
```

Here's the output from the command:

```
DEBUG:  database system was shut down at 2002-11-22 15:33:53 Central Stan
DEBUG:  checkpoint record is in pg_xlog/0000000000000000 at offset 2184988
DEBUG:  redo record is at 0/21571C; undo record is at 0/0; shutdown TRUE
DEBUG:  next transaction id: 541; next oid: 16557
DEBUG:  database system is ready
```

At this point the server is running. To shut it down, I issued a Ctrl-C in the command window. The server responded with a "fast" shutdown:

```
DEBUG:  fast shutdown request
DEBUG:  shutting down
DEBUG:  database system is shut down
```

I could also have used pg_ctl with the stop parameter. Okay, after running start.bat again, I was ready to start mucking about with the database! From my minimal previous experience with PostgreSQL on Linux, I knew that I could use the psql client to get access. And sure enough, in the bin subdirectory, there was psql.exe and a bunch of other utilities. I tried

```
psql
```

and was rewarded with

```
psql: FATAL 1: Database "dharms" does not exist in the system catalog.
```

I was reminded that psql defaults to the current user, and a database with the same name as the current user. This makes multiple user administration much easier, particularly if you change permissions so that users can only access their own databases. But I digress. There was no dharms database, so I tried to create one.

```
createdb dharms
```

and got

```
D:\POSTGR~1\bin>createdb dharms
psql: FATAL 1:  user "dharms" does not
```

```
createdb: database creation failed
```

As Jerry Pournelle would say, "Alas." Time to refresh my memory on PostgreSQL security procedures. Clearly I needed to add myself to the user list, and to do that I needed to know the default superuser id. I tried postgresql, I tried root, I tried a lot of things, including pulling my hair out and emailing Val Raemaekers, who I knew had run the beta successfully. Then I ran psql on a Linux box on which someone else had installed PostgreSQL, and listed the databases. The owner of the default databases? It was `postgres` – not *`postgresql`*, which I'd tried, but *`postgres`*. Now I could create my `dharms` user:

```
D:\POSTGR~1\bin>createuser  -U postgres -e dharms
Shall the new user be allowed to create databases? (y/n) y
Shall the new user be allowed to create more new users? (y/n) y
CREATE USER "dharms" CREATEDB CREATEUSER
CREATE USER
```

I did hear back from Val, who said "I installed pgadminII which I was using to administer postgresql on freebsd, just pointed it to localhost with my windows 2000 username and password, and it connected with no problems at all." So you may not encounter the same problem I did.

With the dharms user in place I was able to run psql without any parameters, and have it default to my user id as the user and database name:

```
D:\POSTGR~1\bin>psql
Welcome to psql, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help on internal slash commands
       \g or terminate with semicolon to execute query
       \q to quit
```

Take careful note of this first message. One, it tells you how to get out of psql – by using the \q command. And two, it tells you that there are a bunch of internal, non-SQL commands you can run, and you can find out about those with the \? command:

```
dharms=# \?
 \a             toggle between unaligned and aligned output mode
 \c[onnect] [DBNAME|- [USER]]
                connect to new database (currently "dharms")
 \C TITLE       set table title
 \cd [DIRNAME]  change the current working directory
 \copy ...      perform SQL COPY with data stream to the client host
 \copyright     show PostgreSQL usage and distribution terms
 \d TABLE       describe table (or view, index, sequence)
 \d{t|i|s|v}... list tables/indexes/sequences/views
 \d{p|S|l}      list access privileges, system tables, or large objects
 \da            list aggregate functions
```

```
 \dd NAME       show comment for table, type, function, or operator
 \df            list functions
 \do            list operators
 \dT            list data types
 \e FILENAME    edit the current query buffer or file with external editor
 \echo TEXT     write text to standard output
 \encoding ENCODING  set client encoding
 \f STRING      set field separator
 \g FILENAME    send SQL command to server (and write results to file or |pipe)
 \h NAME        help on syntax of SQL commands, * for all commands
 \H             toggle HTML output mode (currently off)
 \i FILENAME    execute commands from file
 \l             list all databases
 \lo_export, \lo_import, \lo_list, \lo_unlink
                large object operations
 \o FILENAME    send all query results to file or |pipe
 \p             show the content of the current query buffer
 \pset VAR      set table output option (VAR := {format|border|expanded|
                fieldsep|null|recordsep|tuples_only|title|tableattr|pager})
 \q             quit psql
 \qecho TEXT    write text to query output stream (see \o)
 \r             reset (clear) the query buffer
 \s FILENAME    print history or save it to file
 \set NAME VALUE  set internal variable
 \t             show only rows (currently off)
 \T TEXT        set HTML table tag attributes
 \unset NAME    unset (delete) internal variable
 \w FILENAME    write current query buffer to file
 \x             toggle expanded output (currently off)
 \z             list table access privileges
 \! [COMMAND]   execute command in shell or start interactive shell
```

You won't use most of these commands on a day to day basis, but a few are particularly useful. One is the list databases command, \l:

```
dharms=# \l
  List of databases
   Name    |  Owner
-----------+----------
 admin     | admin
 template0 | postgres
 template1 | postgres
(5 rows)
```

To connect to a database, use the \c command. And once you're connected, you can use \d to list or describe the available tables, indexes, sequences, and views. There are also commands to modify the output of SELECT statements, write results to files, work with large objects, set variables, and more. I'll discuss many of these in upcoming articles.

Summary

With the pending release of a native Windows version of Postgresql, and the immediate

availability of a beta, one of the biggest obstacles to Windows developers using this popular and robust SQL database has been removed. Next time I'll look at basic SQL table creation, and connect to PostgreSQL with Clarion using ODBC.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995). His most recent book is JSP, Servlets, and MySQL, published by HungryMinds Inc. (2001).

Reader Comments

Add a comment

Securing Remote Database Connections With SSH Tunneling

by David Harms

Published 2002-11-28

I've mentioned several times on the newsgroups that I regularly access remote databases across the Internet. When I only had to do this intermittently I just used a plain, unencrypted connection, but now that I do so on a regular basis, I encrypt the connection with Secure Shell (SSH) tunneling. SSH is basically a secure Telnet, and SSH Tunneling is simply a way of piggybacking a database connection on that secure telnet connection. In this article I'll provide some background on the technique, and describe my implementation.

First, a little background. The SSH protocol was created by Finland's Tatu Ylonen as a safe replacement for the "remote" Unix commands, such as rlogin and rsh, which used unencrypted communication. There are both free and commercial implementations of SSH – the free stuff tends to run on Linux/Unix (e.g. http://www.openssh.com) and the commercial versions typically run on Windows or Linux/Unix (e.g. http://www.ssh.com).

To use SSH you need an SSH server program and an SSH client program, and of course they communicate using the SSH protocol. If you're connecting to a Linux server running PostgreSQL or MySQL, then there's a very good chance sshd, the SSH server, is already running. If you don't have SSH on your server, then you'll need to investigate the available implementations for your server platform.

The protocol

The current SSH protocol is 2.0; the 1.x protocols are still supported by most servers and clients, but for the best security you really should use the 2.0 protocol, and keep up with the latest patches.

As SSH stands for Secure Shell, the first and most obvious use of SSH is to let you securely

run a command shell on a server, from a remote client, typically using port 22. Since I administer my servers, I use the command shell to do routine maintenance, run a local (to the server) database client for interactive queries, and so forth. It is also possible, however, to tunnel connections between other ports across the SSH connection, whether or not you use the shell capability.

Tunneling through SSH

If I want to connect to a MySQL server from a remote site, with or without SSH, I'll use the MyODBC driver. Figure 1 shows a typical MyODBC configuration window.

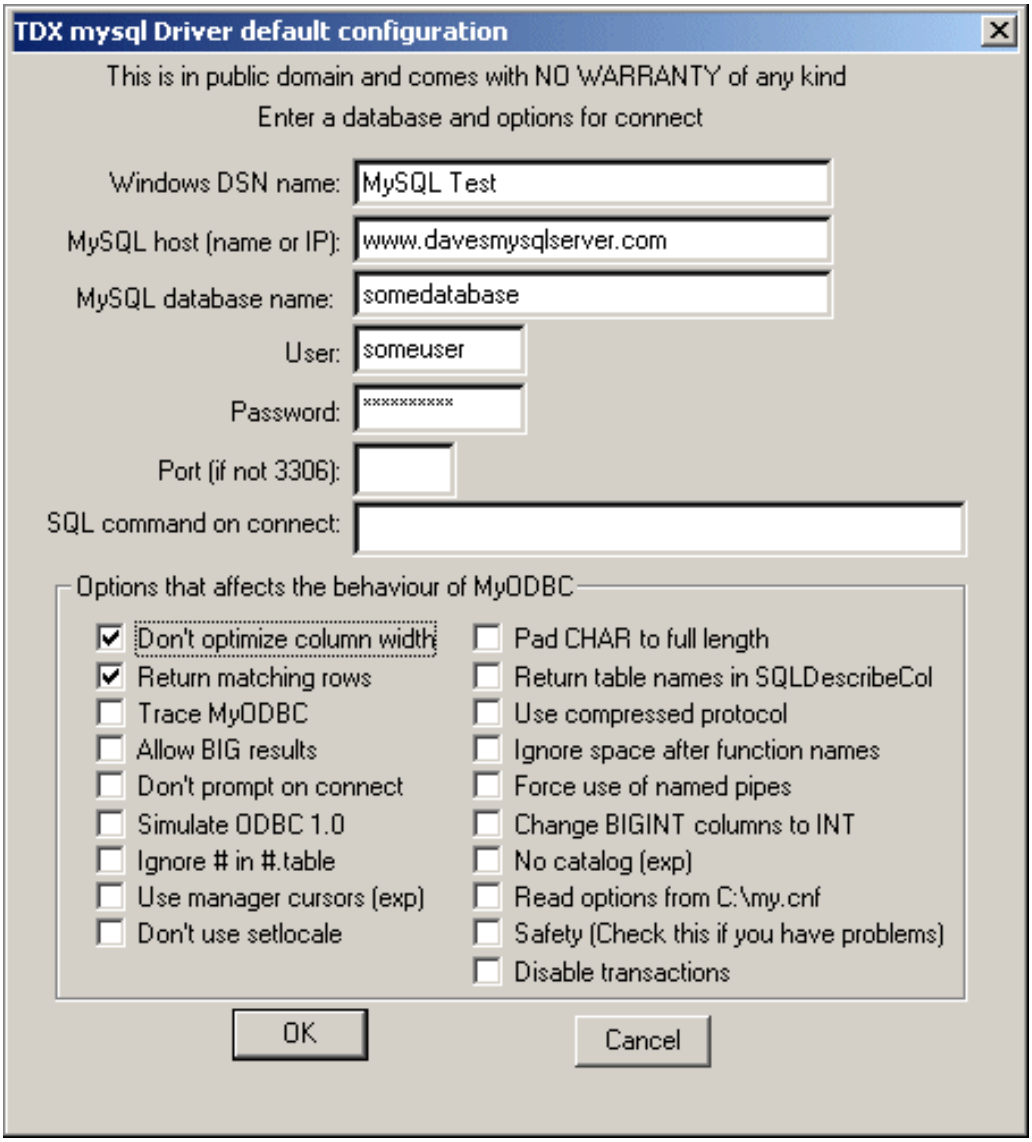


Figure 1. The MyODBC configuration dialog.

Note that the port number defaults to 3306, the standard for MySQL servers. The connection between client and server takes place, in this example, across the Internet, between port 3306 on the server and whatever port happens to be handy on the client (but typically one of the unprivileged ports, i.e. higher than 1024).

EnTunnel

To tunnel the MySQL connection through an SSH connection, I need to tell my SSH client the port numbers to use for the client end and the server end of the tunnel. Just about any SSH client has the ability to tunnel connections this way. For years I've used VanDyke's SecureCRT SSH client to administer my Linux servers, and I've also used it for tunneling database connections. But VanDyke has another, less expensive product called EnTunnel (US$59), which doesn't give you the shell capability but does handle the tunneling very nicely. This makes EnTunnel a good choice for deployment to your clients' machines - progressive discounts apply, so the more you buy, the cheaper the per-license cost, and in most cases you don't want to give your users the opportunity to run a command shell on the server anyway. Even if you don't use Entunnel, the following description will give you an idea of just how easy it is to use tunneling.

EnTunnel runs in the system tray – to configure it, double click on its system tray icon, or right click and choose **Connections** from the context menu. You'll see a Connections window similar to that shown in Figure 2.

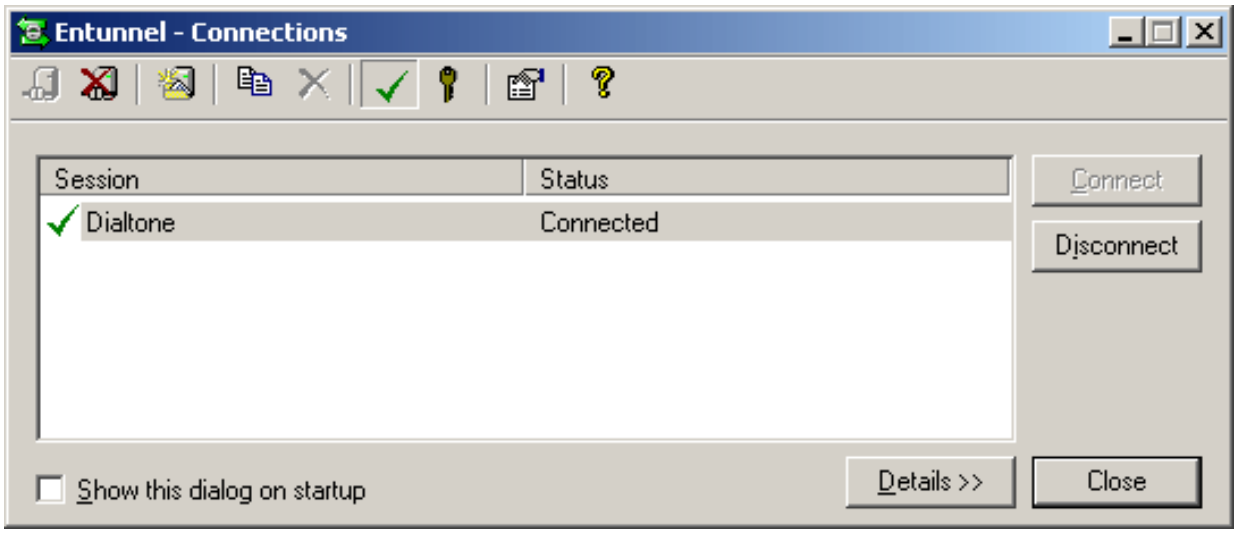


Figure 2. The EnTunnel Connections window, with one connection listed.

To create a new connection, click on the **Create a New Session** icon (third from left) or right click in the list box and choose **New Session** from the context menu. You'll see the dialog shown in Figure 3.

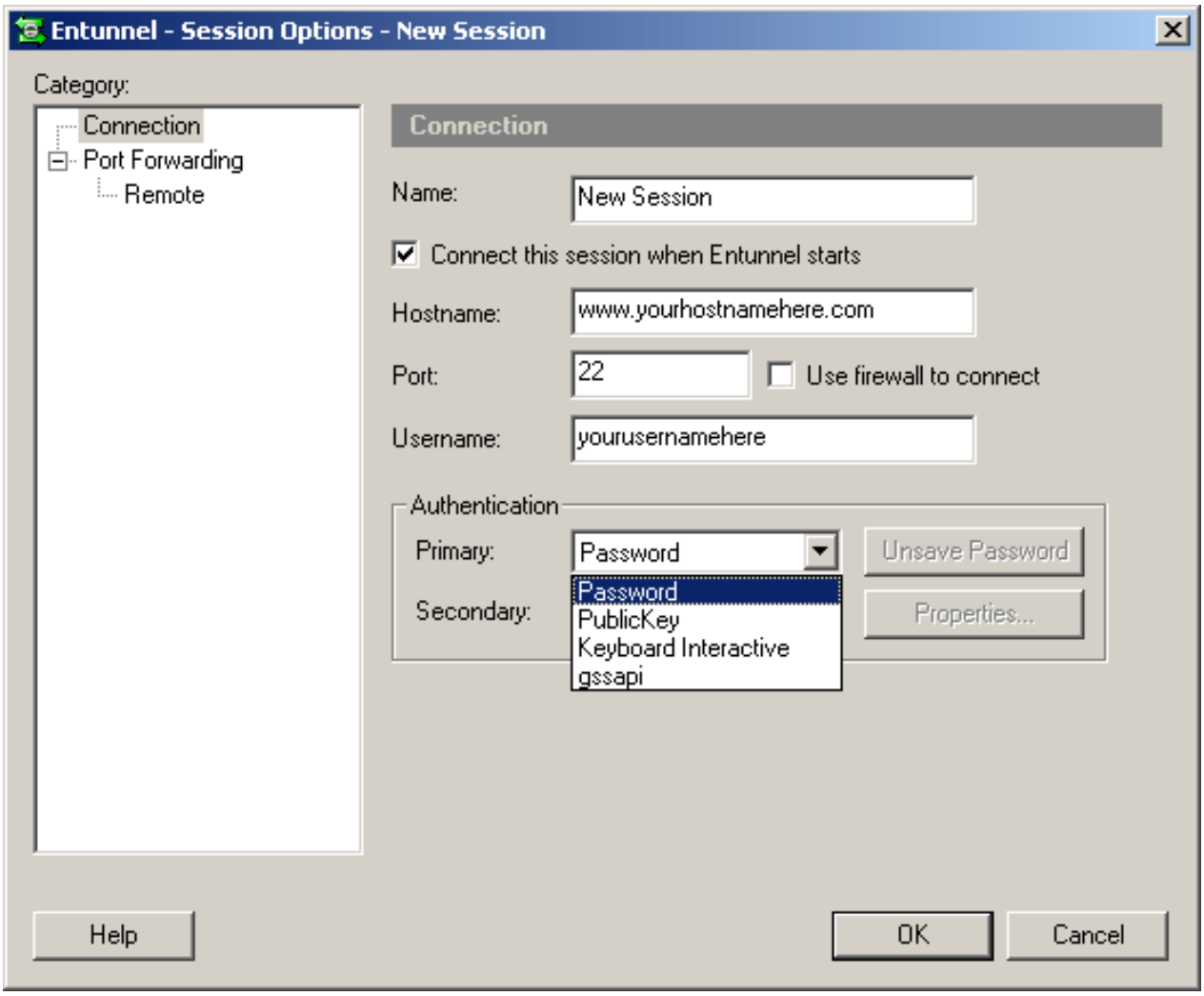


Figure 3. The Create a New Session dialog

The session settings determine how you connect to the SSH server. You need to specify the host name or IP address, the user name you use to connect, and an authentication method. Note that authentication isn't the same thing as encryption – whichever method you use to authenticate yourself to the server, the connection is still encrypted.

If you choose password authentication, then all you need to connect is your username and password (and, of course, the name of the server). This is the lowest level of authentication – if someone can easily guess your username and password, then you don't have much security. Of course your SSH server has to be configured to allow password authentication, which may not be the case.

PublicKey authentication involves a public/private key combination. These are small text files which, in combination, uniquely identify you. The idea is that you distribute a public key to the world at large, but you keep the private key all to yourself. Any message encrypted with the public key can only be decrypted by the private key. EnTunnel has a public key assistant (just click on the **Authentication Properties** button) which will create the key pair for you – you then upload the public key to the server. And as long as you carry your private key and some

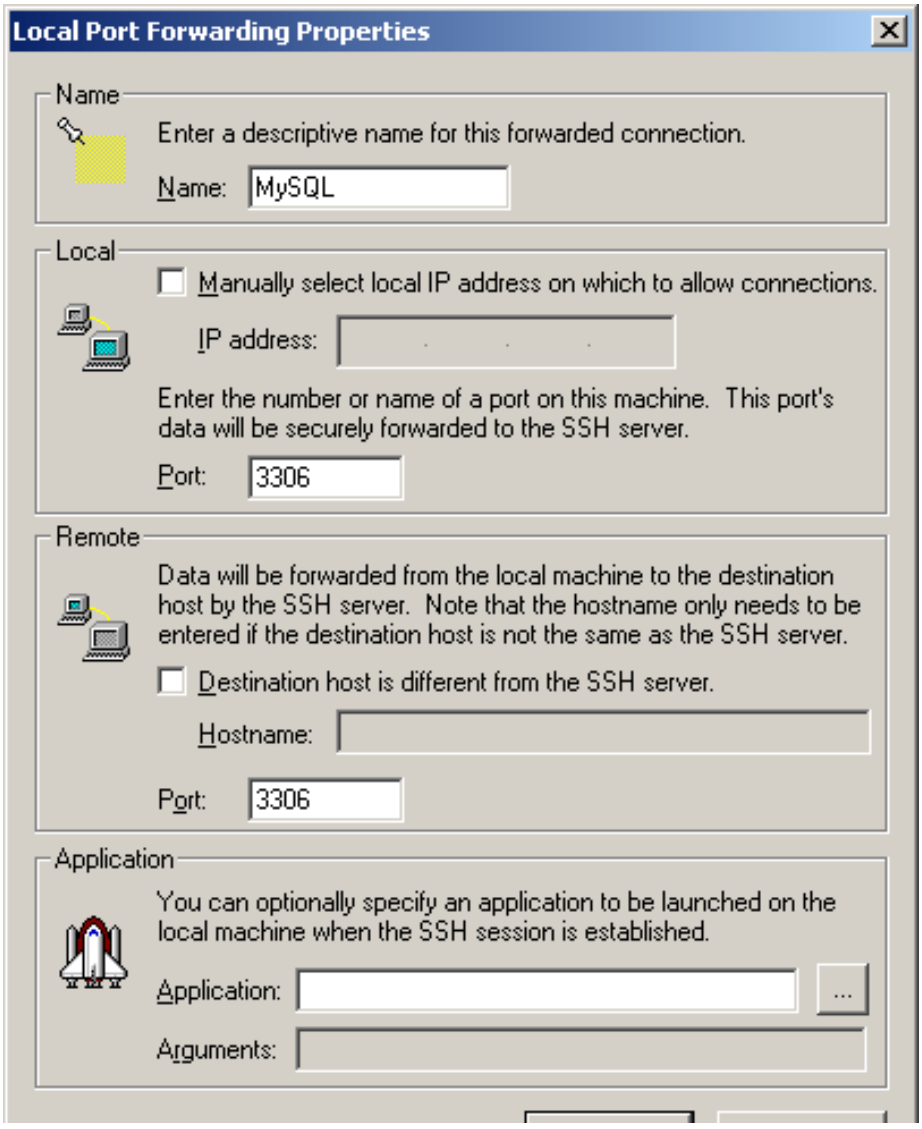
sort of SSH client with you (I suppose one of those USB "key" storage devices would be appropriate) you can connect to your server from anywhere.

Keyboard interactive authentication simply means that you have the option to respond interactively to the server's request for authentication, and gssapi is Kerberos authentication, which I won't go into because I don't know anything about it!

Once you've established and saved your connection settings, you can click on the Connect button from the Connections window and make sure that you are, in fact, able to establish a session with the server. Of course you're not doing anything on the server yet, because you haven't told EnTunnel to forward any ports.

Forwarding ports

To forward a port, right click on the session in the **Connections** window and choose **Properties** from the context menu. Select **Port Forwarding** in the **Category** tree, and click on the **Add** button for the **Local** (not Remote) **Port Forwarding** list. You'll see the dialog shown in Figure 4.



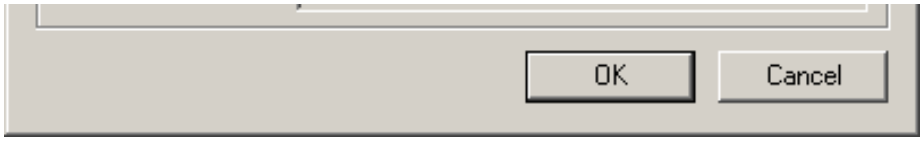


Figure 4. Forwarding a port

Choose a name for the forwarding settings, and the local and remote ports. In Figure 3 I've shown port 3306 (the MySQL default) as the port for the local and the remote sides of the connection. The critical one here is the remote side – that *must* be the port the MySQL server is listening on. The local port can be any valid number (although you may want to avoid the privileged ports), as long as the MyODBC data source is configured to talk to that port also. Figure 5 shows a MyODBC DSN configured to use the tunneled connection. In this example, I'm using 3306 on the local side as well, so I can leave the port number blank and use the default value.

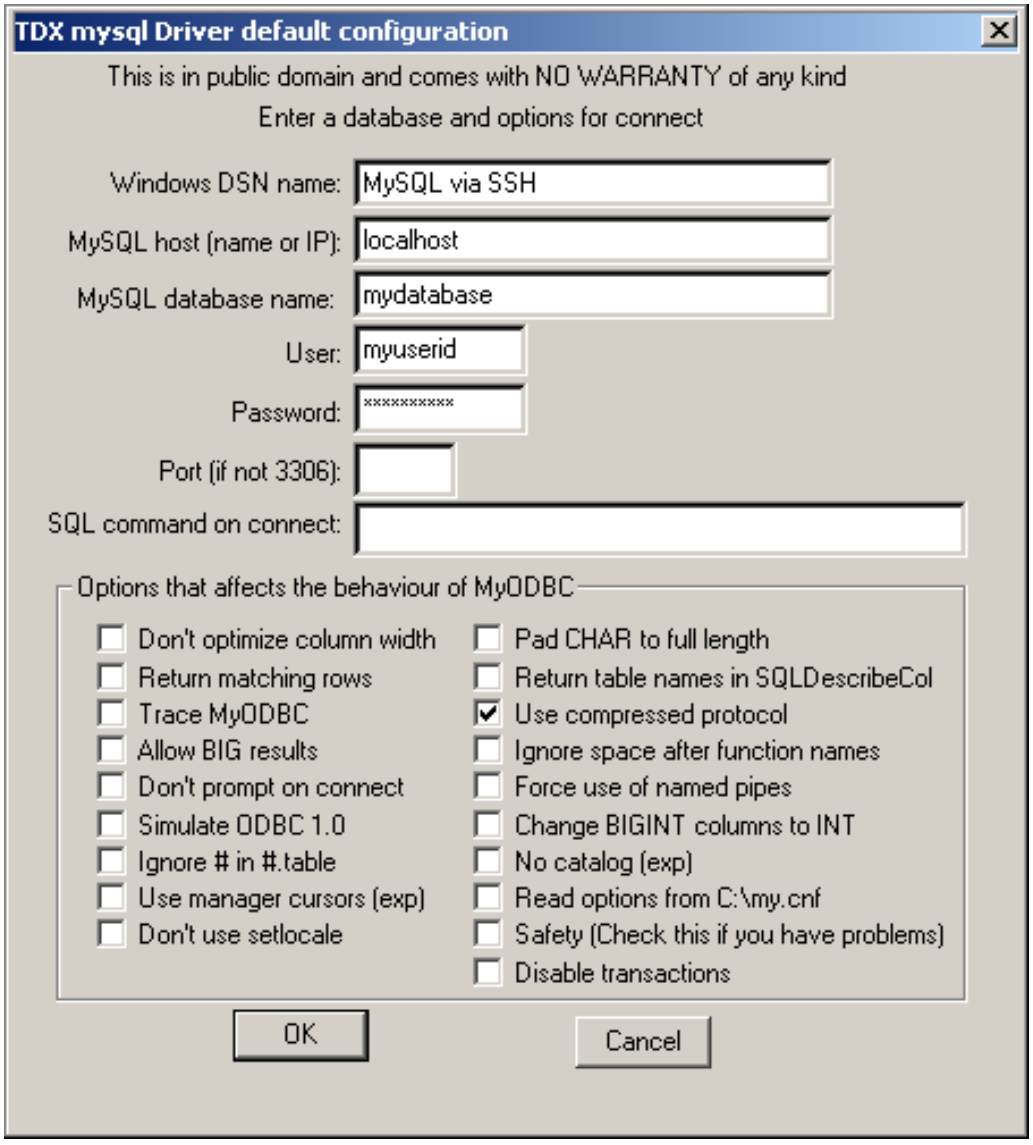


Figure 5. Configuring a MyODBC DSN for tunneling.

Fire up your application, and it will connect to the SSH client on the specified port. The client will encrypt the data and deliver it to the server, and vice versa. You're in business!

Summary

All you need to securely tunnel a database connection across the Internet is an SSH server (freely available on Linux machines), an SSH client, and an ODBC (or other) driver that can connect to a specified port on a specified server. You set up your ODBC data source to connect to a specified local port, and you tell your SSH client to forward that local port to a specified port on the server. You can also use port forwarding to encrypt email protocols and just about any other service with the (usual) exception of FTP.

Port forwarding is an easy and effective way to encrypt database connections, and in many cases it's also the cheapest option available, especially if you're connecting to a database server on a Linux/Unix box and you can use one of the free SSH servers. I can also recommend EnTunnel as an effective and inexpensive tunneling client for Windows.

David Harms is an independent software developer and the editor and publisher of Clarion Magazine. He is also co-author with with Ross Santos of Developing Clarion for Windows Applications, published by SAMS (1995). His most recent book is JSP, Servlets, and MySQL, published by HungryMinds Inc. (2001).

Reader Comments

Add a comment

Clarion SOAP Revisited

by Erik Pepping

Published 2002-11-28

One of our recent endeavors brought us (as RADventure) into Microsoft's .NET world. Soon we were faced with writing a .NET Web service, in C#, to solve a specific problem. The client programs, however, were all written in Clarion. How could we get Clarion to talk to the C# web service?

We knew that we could access the web service using the Simple Object Access Protocol (SOAP). Could Clarion do this? Help!! Panic struck, but then someone remembered that Brian Staff had written a small article in Clarion Magazine on calling a SOAP server from a Clarion client. With that start, and some reading on the web, we soon got it working. To spare you even that time I will present a small example.

The web service

First, the web service. A web service is an object residing on a Server in a network. You can create a web service with all kinds of tools, like Java, C++, Visual Basic 6, or with C# in the .NET framework. A web service can be called from anywhere as long as the network can reach the server which hosts the service .

It's fairly easy to create a web service in Visual Studio .NET.

First, create a new project, indicating you want to create an ASP.NET web service. Figure 1 shows the New Project create dialog in Visual Studio.

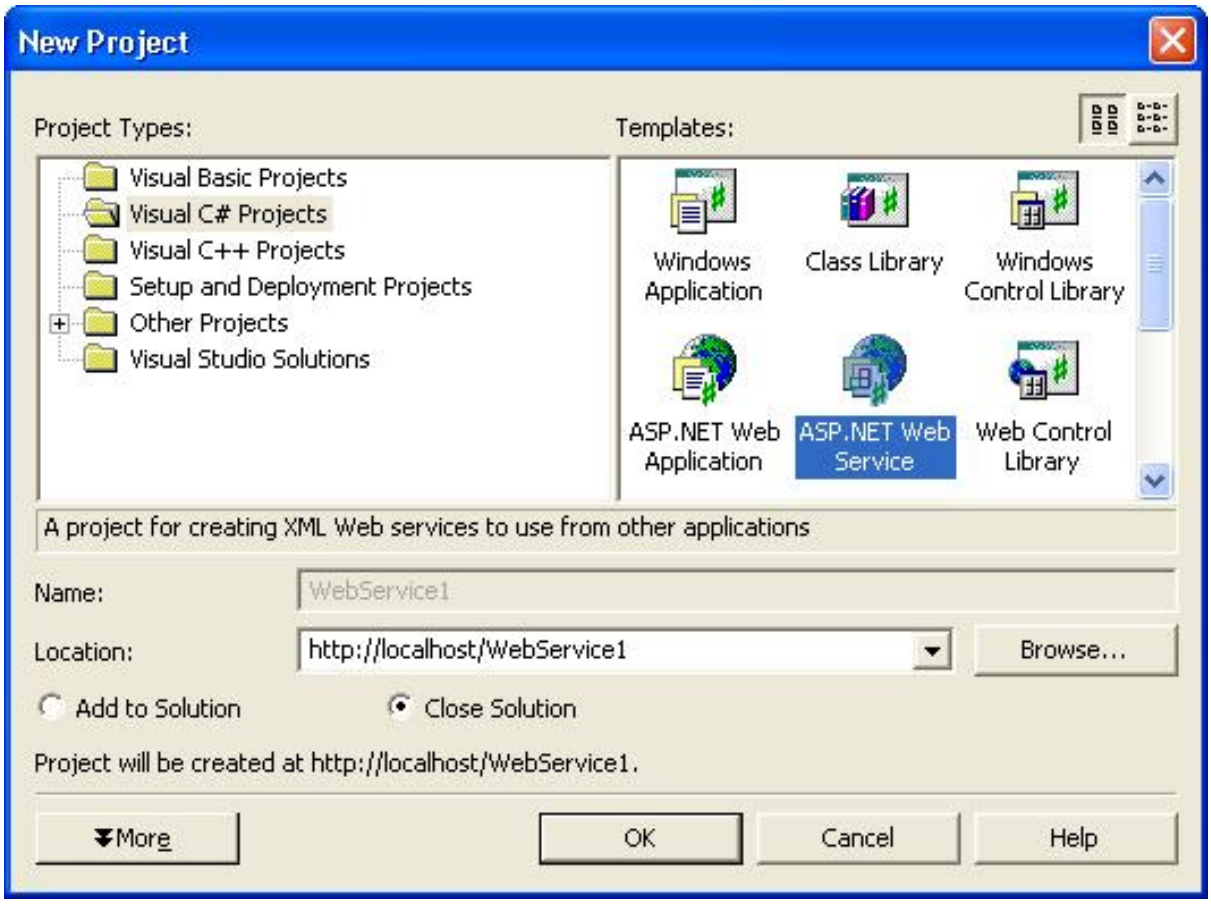


Figure 1. Creating a new web service in Visual Studio .NET.

When you click **OK**, Visual studio builds a skeleton web service to which you can add your own methods. We created three methods for our web service: two test functions (`HelloWorld` and `AddAToB`), and one, called `GetCustomerAddress`, to retrieve customer data residing in our Oracle database. For simplicity the data is returned as a normal string, not as XML (although XML is more common, and just as easy to implement using the .NET XML classes).

Here is the completed source code:

```
// Namespaces to be included to make the
// webservice and Oracle data access work
using System;
using System.ComponentModel;
using System.Data;
using System.Web;
using System.Web.Services;
using System.Data.OleDb;
namespace MyWebservice  // a namespace is a way of
                        // grouping classes together
{
    ///
    /// Summary description for Service1.
    ///
    [WebService(Namespace="http://127.0.0.1/webservices/")]
```

```
public class Service1 : System.Web.Services.WebService
{
    // this is the constructor, has the same name
    // as the Service1 class. service 1 inherits
    // from the framework webservice class
    public Service1()
    {
        // CODEGEN: This call is required by the ASP.NET
        // Web Services Designer. The initialize component
        // exists in the webservice class
        InitializeComponent();
    }
    //Required by the Web Services Designer
    private IContainer components = null;

    ///
    /// Required method for Designer support - do not modify
    /// the contents of this method with the code editor.
    ///
    private void InitializeComponent()
    {
    }
    /// Clean up any resources being used.
    protected override void Dispose( bool disposing )
    {
        if(disposing && components != null)
        {
            components.Dispose();
        }
        base.Dispose(disposing);
    }

    // The HelloWorld() example service returns
    // the string Hello World
    // The [webmthod] is a compiler directive telling the
    // compiler this method is to be exposed as a external
    // webservice callable function
    [WebMethod]
    public string HelloWorld()
    {
        return "Hello World";
    }
    [WebMethod]
    public int AddAToB(int a, int b)
    {
        return a + b;
    }
    [WebMethod]
    public String GetCustomerAddress(int CustomerId)
    {
        OleDbConnection objCon = new OleDbConnection
           ("Provider=MSDAORA.1;Password=ers;User ID=ers;↵
                Data Source=radv;Persist Security Info=True");
        string strSQL= "SELECT Naam, Straat, Postcode, Plaats ↵
                from Relatie,adres where Relatie.c_relatie=↵
                Adres.C_relatie and Relatie.C_relatie=" + CustomerId;
```

```
            string retstring;
            try
            {
                objCon.Open();
                OleDbCommand cmd = new OleDbCommand( strSQL, objCon);
                OleDbDataReader reader = cmd.ExecuteReader();
                reader.Read(); // Advance to the one and only row
                retstring = reader.GetString(0) + "\n" + reader.GetString(1)
                  + "\n" +reader.GetString(2) +  " " + reader.GetString(3);
            }
            catch
            {
                throw;
            }
            finally
            {
                objCon.Close();
            }
            return retstring;
        }
    }
}
```

The source code is fully documented and should be easy to read. In the end, all that matters is that there are three methods that have become callable methods within the webservice named Service1. This class inherits from `System.Web.Services.WebService`, which is the .NET framework class that hides all the intricacies of the webservice and the webserver, the communication protocols, etc.

Figure 2 shows the result of connecting to the service with the browser; the service itself is running in Visual Studio .NET.

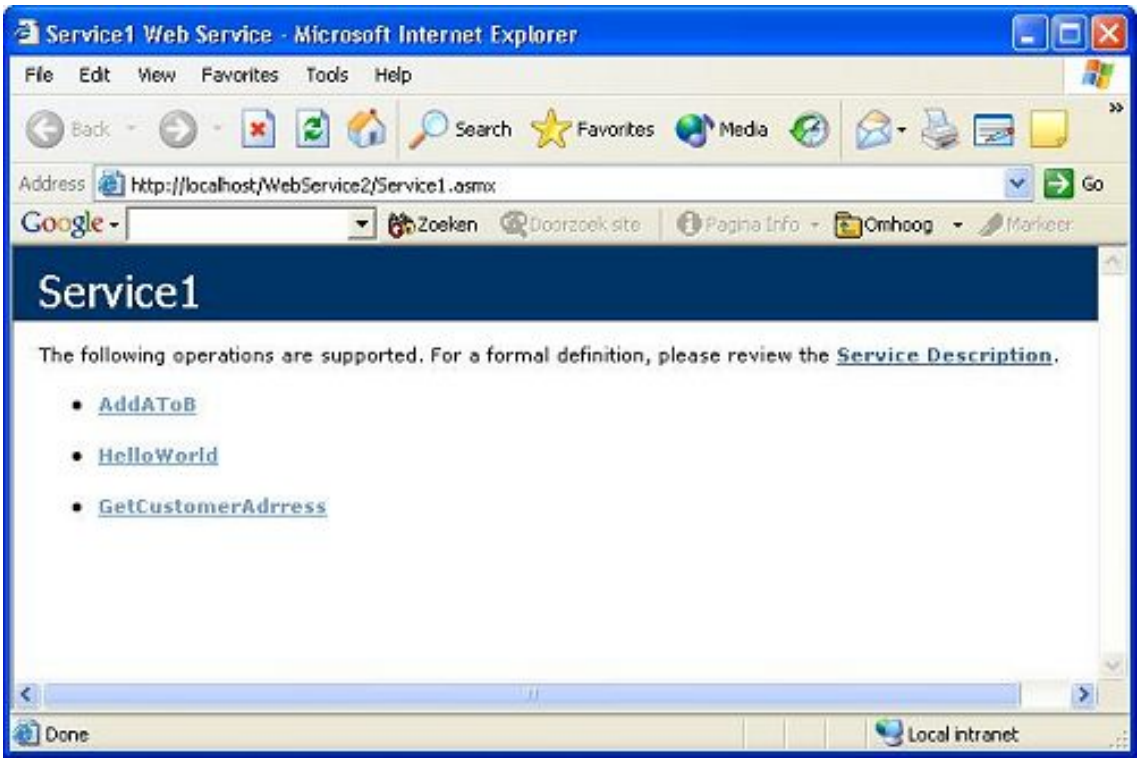


Figure 2. The Service1 web service

As you can see, the generated page lists the methods that are programmed in the C# source code.

You can call method **AddAToB** by clicking on in its link. The browser will now ask for the two input variables, as shown in Figure 3..

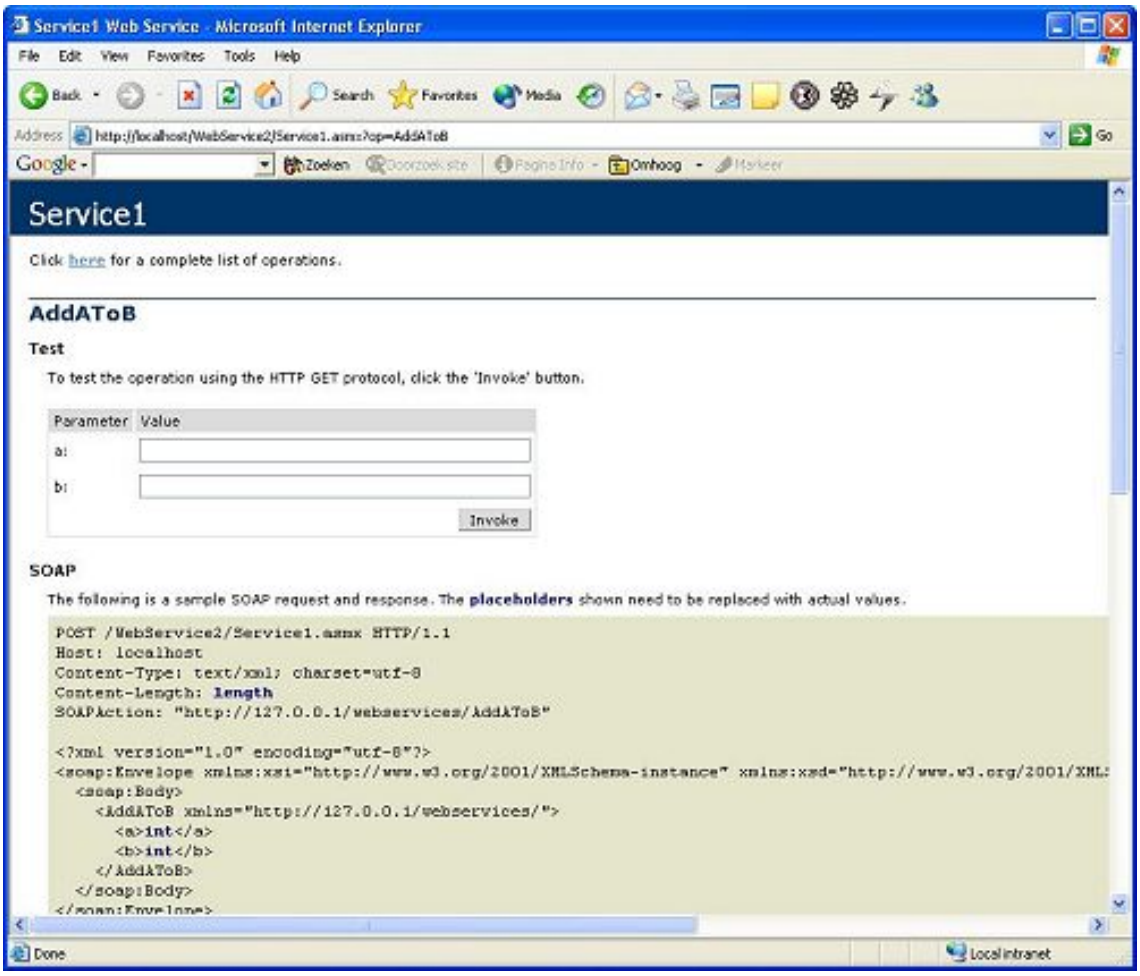


Figure 3. Testing the AddAToB function.

If you enter **a** and **b** as 3 and 5 and click on the **Invoke** button, the web service returns the result in XML, as shown in Figure 4.

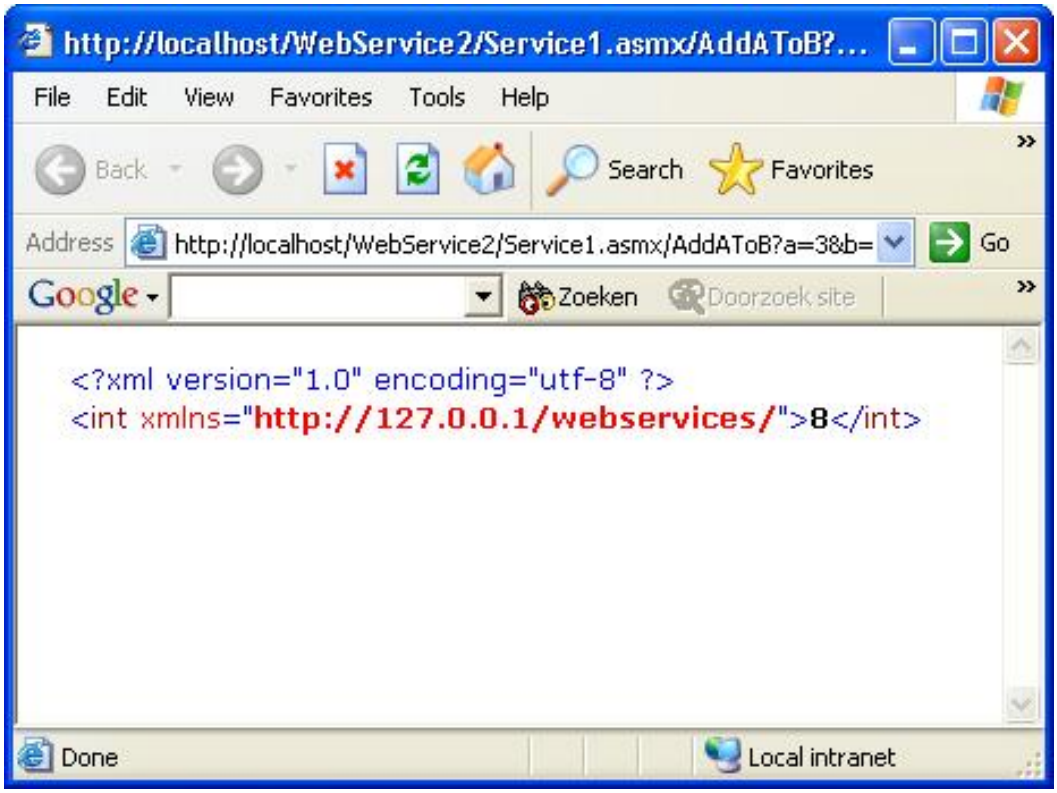


Figure 4. The XML result of AddAToB

As you can see (no surprise here) the result is 8. You can probably guess how the `HelloWorld` method works, so I won't bother you with that description. The third method, `GetCustomerAddress`, is, however, interesting and of real use. It retrieves customer data depending on the customer id passed to the method as a parameter. The function uses Oledb to log into the Oracle database, issue a `SELECT` statement with the passed customer ID, and return a string containing name, street and city to the caller. (Oledb is the follow-up technology slowly replacing ODBC. Formally it is called ADO .NET. Look for ADO classes in the upcoming Clarion 5.6.)

The Clarion code

How to invoke this from Clarion ? Well, it is quite simple. The following code uses Jim Kane's `Olectl` class (which employs late binding) to create a SOAP client, initialize the client, and then call each of the service's methods in turn, displaying the results. Not surprisingly, the SOAP client code is much shorter than the .NET service code:

```
soapclient  program
_abcdllmode_  equate(0)
_abclinkmode_ equate(1)
  map
  end
Include('oletcl.inc')
!for soap demo
OClient      &oleTClType
Par1         Cstring(100)   ! Parameter web service name
```

```
Textx        CString(100)   ! Results
A            short(3)       ! A parameter for add A to b method
b            short(5)       ! B parameter for add A to b method
 code
 ! Define a new ole class instance
 oClient &= NEW oleTClType
 ! initialized tthe soap ole object
 oClient.init('MSSOAP.SoapClient',0)
 oClient.CallMethod('ClientProperty("ServerHTTPRequest",True)',1)
 par1 = 'http://localhost/WebService2/Service1.asmx?WSDL'
 oClient.CallMethod('mssoapinit ("' & CLIP(par1) & '" , "", "" ) ',1)
 ! Call the hello world method of the webservice
 textx = oClient.Getprop('HelloWorld()')
 ! display the result
 Message(textx)
 ! Call the addAToB method of the webservice
 textx = oClient.Getprop('AddAToB(' & a & ',' & b & ')')
 Message(textx)
 ! Call the getcustomeraddress methode of the webservice
 textx = oClient.Getprop('GetCustomerAddress(4)')
 ! Display the result
 Message(textx)
 oClient.kill ! Kill the object
```

Let's explore this a bit. The first line creates a new instance of `OletCltype`, which is Jim Kane's class for OLE late bound processing. In `the .Init` method the `MSSOAP.SoapClient` descriptor indicates the kind of OLE object to be created. Then the code calls the `ClientProperty` method indicating that the call to the remote object should be done using the HTTP protocol. Then a particularly important parameter is passed: `'http://localhost/WebService2/Service1.asmx?WSDL'`. This points to the local webserver and a virtual directory for IIS called `WebService2`. The actual service is `Service1.asmx`, and the `WSDL` parameter to the request asks the service for its Web Services Definition, an XML document (created on the fly) which describes, among other things, the methods and parameter data types available.

To call a method and return a result all you have to do is use the `Getprop` method with the required method name and parameters. But be careful - C# is case sensitive, so the method names must be exactly as expected.

The result is three message boxes, as shown in Figure 5.

Figure 5. The results of the three method calls.

We were surprised how easy it is to use SOAP to call a web service created with the .NET C# language. Remember that you need the SOAP client. It is installed already in windows XP, or if you have installed the .NET framework. Otherwise you can download the SOAP client from Microsoft.

Download the source

Erik Pepping began programming in Clarion with the DOS 2.1 product, and is still on the Clarion path. Two times he tried to escape by diverting to Powerbuilder and Visual Basic, but he tired of the tedious coding and came back to Clarion programming, where he has been ever since. Erik has a degree in Information Sciences from a technical university in the Netherlands. He recently became managing director of RADventure, a merger between his part-owned company and Advantage, the previous distributor of Clarion in the Benelux. Erik, Andre van der Weijden, and the well-known Rens brothers (Arie and Peter) are co-owners of RADventure. Erik has won lasts year Dutch RADrace together with a colleague, Peter Rakke. Erik manages Clarion projects, occasionally teaches advanced Clarion courses, and has spoken at several international congresses in the U.S., Australia, and Europe, generally on object oriented programming or SQL.

Reader Comments

Add a comment