Marex Returns Home to BASIS - A Personal Journey

By Kurt Williams

Rewind to 1979. I wrote my very first line of Business BASIC code on an MAI BasicFour 410. It was exciting to control a video screen even if it was just controlling where characters went on the screen. Before Business BASIC, I wrote programs for batch mode computers that took input from decks of punch cards and sent output to printers, more punch cards, magnetic tape, and, if I was lucky, a magnetic disk! Using a video screen to communicate with a user was a huge advancement.

> *Editor's note:* If these first legs of Kurt's personal computer and programming journey ring familiar to you with punch cards, magnetic tape, and MAI BasicFour, you will especially enjoy the STARTUP exhibit featured at TechCon2007. *STARTUP: Albuquerque and the Personal Computer Revolution* (www.startupgallery.org) is a gift from Microsoft co-founder Paul Allen to Albuquerque. It is the first-ever exhibit dedicated to the history of the PC and software, much of which unfolded in Albuquerque.

Fast-forward twenty years. After much hard work developing and maintaining Business BASIC software and holding diverse positions including some with BASIS International Ltd. (BBx Product Manager and member of the Engineering Team), I left the Business BASIC world entirely. I accepted a position running an IT department for an electrical products distributor that managed a geographically-large distributed wide area network with several hundred users. There, I used several technologies including Java and SQL.

As my Java development skills and SQL knowledge grew, new object-oriented design, integrated development environments, and relational database management tools made programming in Business BASIC seem like ancient history. Capitalizing on this new knowledge, I later started a software contracting business to focus on developing business applications using Java, MySQL, and other technologies. Marex Services was born. Now several years later, Marex Services has built a healthy client base of satisfied customers.

A few months ago, BASIS asked Marex Services to develop some code for them using their most advanced technologies. I was skeptical. I was a Java developer now but they persisted. They insisted that the new BASIS technologies competed quite well against other development tools and then challenged me to take on the task and prove them right. I accepted the challenge.

Re-entering the BASIS World

Upon initial examination, nothing had changed in the BASIS world. All the old language features, database functionality, and development tools were still in use. But with further examination, I discovered that many more tools were available; a new Integrated Development Environment (IDE) built on the already familiar Net Beans open source project, a plethora of new DBMS file types and most significantly, an object-oriented Business BASIC backed by a powerful BBjAPI (Application Programming Interface) as well as a new SQL engine.

Discovering the BASIS IDE

Soon, I experienced just how powerfully the BASIS IDE brought all the disparate development tools together. I edited, compiled, and ran my programs all in the IDE's debugger. When my compilation errors occurred, they displayed in the compiler output window and a quick double-click on any of these errors opened the source file and highlighted the offending line. When my changes were complete and I was ready to save them to the system, I used the IDE's CVS client to commit the changes to my CVS repository.

The IDE contains several plug-in modules that were very handy. The data dictionary plug-in allowed me to access the data dictionary either locally or remotely and get answers to questions quickly. Similarly, I could easily add or edit new data dictionary definitions. The FormBuilder IDE plug-in provided me with a visual layout tool for graphical user interfaces. I could quickly build and maintain the GUI forms that I used in my business application. The AppBuilder module, integrated with the FormBuilder module, provided an easy-to-use framework for coding event handlers for my newly created forms. This framework was within the same IDE interface so it managed the event queue to deliver on its RAD promise.

What I discovered was that the IDE is as strong a work bench for my development projects as any IDE that I have ever used. In fact, it is light years beyond any BBx development tool I used years ago.

Discovering the BBjAPI

In the days when I developed in Java and approached a problem, I thought about what resources the Java API provided that I could use to solve this problem. Now working in BBj®, my thought process was similar, but expanded. I first thought about solving the problem using the BBjAPI and most of the time I found the

continued...

Kurt Williams
Principal
Marex Services

solution. However, if I needed a bit more help, I moved outward to the Java API. With this synergy, the two working together always provided a solution. There really were no limits!

One of my greatest discoveries about the BBjAPI was how much time it saved me. For example, if I was using PRO/5® and had a list button on my form that I needed to load from a database table, I would have opened the file, read each record that met my criteria from the file, and placed it into the list button control. Using the record set capabilities in the BBjAPI, I reduced all those steps to the following:

```
connectStr$=" jdbc:basis:ourServer:2001?database=dbName&user=admin&password=mypwd"
sqlStr$="SELECT StateAbbr FROM States Order By StateAbbr"
statesRecordSet! =  BBjAPI().createSQLRecordSet(connectStr$, mode$, sqlStr$)
statesListButton!.fillFromRecordSet(statesRecordSet!,"StateAbbr")
```

The **connectStr$** variable was the information the SQL Engine needed to connect to **ourServer**. It further said to use port **2001** and open the database **dbName** as the user named **admin** with the password **mypwd**. The **sqlStr$** entry was the SQL SELECT statement needed to load the **States** list button. The third line created an SQL record set object using **connectStr$** and **sqlStr$**. The fourth line filled the BBjListButton object from the BBjRecordSet object created in the previous three.

This BBjAPI approach was straightforward and easy. There was no need to build a read loop loading each individual record read into the list button and no need to handle the error when the loop gets to the end of the file. The **fillFromRecordSet** method called in the BBjListButton object handled all these functions.

Considering all these features, I found the BBjAPI was a rich toolbox of objects and methods for managing the interfaces and processes that made up my business applications.

Discovering the SQL Engine

The BBj SQL Engine was a first rate SQL processor for managing my applications data whether it resided in the BBj DBMS files or in a database manager such as MySQL. Much to my surprise, I found that I could do everything with the BBj SQL Engine that I was accustomed to doing with SQL tools outside of BBj.

For example, I needed to develop a list of serial numbers registered to a given customer that also had an end user record registered to the serial number. Using a traditional Business BASIC approach, I would have written something like this:

```
read (serialNumber,key=customerNumber$,knum=1,dom=*next)serialNumRec$
readLoop:
        read record(serialNumber,end=doneLoop)serialNumRec$
        if serialNumRec.customerNumber$<>customerNumber$ then goto doneLoop
        read record(endUserLink,key=serailNumRec.serialNum$,dom=readLoop)endUserLink$
        read record(endUserData,key=endUserLink.endUserNumber$)endUserData$

        REM .. do whatever we need to do ..

        goto readLoop:

doneLoop:
```

While this worked, this approach was time consuming. However, when I moved the processing to the BBj SQL Engine as follows, it ran much faster:

```
connectStr$= "jdbc:basis:ourServer:2001?database=dbName&user=admin&password=mypwd"
sql$="SELECT T2.SERIAL_NBR, T3.ACTIVE_FLAG, T1.*, T4.INDUSTRY_NAME "
sql$=sql$+"FROM EndUserData T1 "
sql$=sql$+"INNER JOIN EndUserLink T2 ON T1.END_USER_NBR = T2.END_USER_NBR "
sql$=sql$+"INNER JOIN SerialNumber T3 ON T2.SERIAL_NBR = T3.SERIAL_NBR "
sql$=sql$+"WHERE T1.CUSTOMER_NBR='" + customerNumber$ + "' "
sql$=sql$+"ORDER BY T2.SERIAL_NBR"
ourRecordSet!=BBjAPI().createSQLRecordSet(connectStr$, mode$, sql$)

while 1
        rowData! = ourRecordSet.getCurrentRecordData()

        REM .. do whatever we need to do ..

        ourRecordSet.next(err=*break)
wend
```

continued...

With this SQL SELECT, the SQL Engine handled all the checking to see that I was only retrieving end user data linked to serial numbers registered to the customer in question. The BBj SQL Engine was certainly equal to any SQL processor that I had ever worked with.

Reflections

At the conclusion of this challenge, I discovered an impressive set of tools to help the programmer develop robust and reliable code, more rapidly than ever before. The BASIS IDE is equal to the tool I once used to develop Java code and in fact, I can also use it to develop Java code. The BBjAPI provides a rich array of objects for creating business applications. The BASIS SQL Engine is a versatile tool for the manipulation and retrieval of data from BASIS DBMS and foreign databases as well.

BBj and all its associated tools is a world-class development suite.

Now, after working with BBj on several more development jobs, I am convinced, and am the living proof that what BASIS told me is true; BBj and all its associated tools is a world-class development suite. Unlike the book title by Thomas Wolfe that so many freely quote, "You Can't Go Home Again," I left Business BASIC years ago for the Java world and have in fact, returned home. My foundational Business BASIC development experience applied directly to all I learned while working with Java and allowed me to get things done far more quickly with the BASIS IDE. The results of this great journey were highly maintainable and very reliable business applications.

As they say, "home again, home again!"

See also *Confessions of a Language Polygamist*
www.basis.com/advantage/mag-v10n1/confess.html