Using
MultiValue Data
with
ASP.NET OWIN

Also in this Issue

- $6M Date
- UniVerse and Python - It's Here!
- Case Study: MBS helps SA Police Super Modernize
- CSP vs. Snowflakes

INTERNATIONAL SPECTRUM

THE MULTIVALUE TECHNOLOGY MAGAZINE

CONTENTS

From the Inside

Welcome to the International Spectrum 2018 Conference!

For those that are here in Florida, Welcome, Enjoy… I'm sure you and I will be talking throughout the conference. There's a lot of material and plenty of networking opportunities.

For those that were unable to make it, we don't want you to feel left out. The good news is that Spectrum 2018 is actually two events in one: live and on-demand. Attendees get access to both but the on-demand portion is also available for the rest of our community.

You see, during the conference we record what is projected by the speakers along with the audio of the presentations. These recordings then are provided to our conference attendees to view afterwards, on-demand.

Many times, an attendee will see something at the conference they want to implement but don't get a chance to do this until months down the road. For those of you here at the live conference, you get access to the on-demand content as part of your attendance to the live conference.

For those that can't make it, we also offer access to just the on-demand content. So if you are restricted by budget or time, you can still see many of the sessions. This is not as good as attending in person, since you can't ask questions or talk with other attendees but it still provide you with needed educational materials.

Find out more at:

http://www.intl-spectrum.com/conference/

NATHAN RECTOR
President
International Spectrum
nathan@intl-spectrum.com

MISSION STATEMENT *International Spectrum* magazine's editorial mission is to be the premier independent source of useful information for users, developers, and resellers of MultiValue database management systems, open systems business database solutions, and related hardware, software, and peripherals. Published bimonthly, International Spectrum provides comprehensive coverage of the products, companies, and trends that shape the MultiValue marketplace as well as the computer industry at large — helping its readers get the most out of their business computer systems.

twitter.com/intlspectrum   intl-spectrum.com/facebook

BY NATHAN RECTOR

Using OWIN Security
with
MultiValue Data
Part 2

I n my last article, I explained a little bit about what OWIN was, and the basic setup of how to create a connection between an OWIN application and a MultiValue framework. This article is mainly focused on using OWIN's identity/security framework.

Quick Review

ApplicationDBContext — This class does all the work of connecting to the database and making the subroutine calls to return or update data.

ApplicationUser — This class is used to hold information about the user, making it available for the rest of the OWIN Identity system. This class will call the subroutine, in out case SPECTRUM.OWIN.USER, and return a dynamic array of information about the user.

ApplicationUserStore

Now that you have the basis for the ApplicationUser class, you have to implement the ApplicationUserStore class. This class does most of the work

> *The OWIN granularity means you don't have to implement everything...*

connecting the data from the subroutine SPECTRUM.OWN.USER, with the rest of the Identity system.

This is where the OWIN Identity System gets a little ugly. Microsoft, in their infinite wisdom, decided to create a very granular identity system. This specifically addresses the biggest complaint about previous frameworks: They were either too cumbersome to implement and user, or too subtle to customize to individual needs.

The OWIN granularity means you don't have to implement everything they want you to, if you don't need it or to want it. The reality though, is that in order to take advantages of what the OWIN identity framework provides, you pretty much have to implement all of it anyway. This really isn't a big issue since the individual interfaces are actually relatively simple to imple-

ment. And while annoying to have to type in all this code, it does make it easier to write articles that walk you through it.

Interface IUserStore

The only Interface that you are required to implement is the IUserStore(Of IUser) (See Figure 1). It must have the ability to Create, Update, Delete and Find users.

As you look at this you will notice that seems a little too simple. Well, it is. Sometimes we get lucky. IUserStore does not implement any password handling or role management or any ability to do login or out. This will be handled by the next few interfaces.

Two things that I'd like to highlight [*Figure 1*], are the FindByIdAsync and FindByNameAsync functions. They provide OWIN with a way to lookup existing users in your database. The ApplicationUser class does not have a way to do these lookup by default, so you have to implement this lookup yourself. I've provided and example to

```
Public Class ApplicationUserStore
    Implements IUserStore(Of ApplicationUser)

    ...
    Public ReadOnly Property Users As IQueryable(Of ApplicationUser) Implements
IQueryableUserStore(Of ApplicationUser, String).Users
        Get
            Return _Users
        End Get
    End Property
    Private _Users as New List(Of ApplicationUser)

    Public Function CreateAsync(user As ApplicationUser) As Task Implements
IUserStore(Of ApplicationUser, String).CreateAsync
        Return user.SaveAsync
    End Function

    Public Function UpdateAsync(user As ApplicationUser) As Task Implements
IUserStore(Of ApplicationUser, String).UpdateAsync
        Return user.SaveAsync
    End Function

    Public Function DeleteAsync(user As ApplicationUser) As Task Implements
IUserStore(Of ApplicationUser, String).DeleteAsync
        Throw New Exception("User Delete is not supported")
    End Function

    Public Function FindByIdAsync(userId As String) As Task(Of ApplicationUser)
Implements IUserStore(Of ApplicationUser, String).FindByIdAsync
        Return FindUserAsync(DBContext, "Id", userId)
    End Function

    Public Function FindByNameAsync(userName As String) As Task(Of ApplicationUser)
Implements IUserStore(Of ApplicationUser, String).FindByNameAsync
        Return FindUserAsync(DBContext, "UserName", userName)
    End Function

    ...

End Class
```

Figure. 1

show how I approached it using a subroutine [*Figure 2*] to help you implement your own solutions.

Keeping a cached copy of the retrieved information in memory is one of the most important things that we need to be doing each time we access a user from the database. Doing this keeps the application responsive. As we all know, the slowest part of any client server applications is usually the communication with the database itself.

Since this is a memory cached copy, we don't always know how long an item has been in memory and needs to be refreshed. It is always important to keep track of when something is placed in cache so you know when to refresh the information in case it has changed.

I've done this by added an extra property to the ApplicationUser class we built in Part I [Tracey: we need a link to that issue here] called LastRead. Every time an ApplicationUser is read from, or saved to, the database, this property will be set to the current date/time. This allows us to keep track of how old the data is and help us decide

```vbnet
    Public Async Function FindUserAsync(DBContext As ApplicationDBContext, FieldName As
String, FieldValue As String) As Task(Of ApplicationUser)

        ' Create the User Object if not found
        Dim _User As ApplicationUser = Nothing
        For I As Integer = 0 To (_Users.Count - 1)
            If FieldName.ToLower = "id" AndAlso _Users(I).Id = FieldValue Then
                _User = _Users(I)
            ElseIf FieldName.ToLower = "username" AndAlso _Users(I).UserName =
FieldValue Then
                _User = _Users(I)
            ElseIf FieldName.ToLower = "email" AndAlso _Users(I).Email = FieldValue
Then
                _User = _Users(I)
            End If
        Next

        If _User IsNot Nothing Then
            ' Found an item, so check to see if we need to update it or not.
            If _User.LastRead.Add(TimeSpan.FromMinutes(10)) > Now Then
                ' The last time this item has been read was longer than 5 mins ago
                ' reread current information from database
                _User.LastRead = Now
            Else
                ' ReRead the infomraiton
                Await _User.LoadAsync()
            End If

            ' Returns the user information
            Return _User
        Else
            ' Didn't find a user, so look it update in the database
            Dim _RequestItem As String = "FIND.USER"
            _RequestItem = _RequestItem & mvFunctions.AM & FieldName
            _RequestItem = _RequestItem & mvFunctions.AM & FieldValue

            ' Send the data to the server
            Dim _DataItem As String = String.Empty
            Dim _Result As ApplicationDBContext.CallSubroutineResults = Await
DBContext.CallSubroutineAsync("SPECTRUM.OWIN.USER", _RequestItem, _DataItem)
            If _Result.CallError.BooleanValue(1, 0, 0) Then
                ' Error, can't find the information
                Return Nothing

            ElseIf _Result.Data.Item(0).IsNullOrEmpty Then
                ' Nothing returned
                Return Nothing

            Else
                ' Extract the User Id and User Item and create
                ' the User Object
                _User = New ApplicationUser(_Result.Data.Item(0).StringValue(1, 1),
_Result.Data.Item(1))
                _Users.Add(_User)
                Return _User
            End If

        End If
    End Function
```

Figure. 2

whether our application should reload the data or not.

I do this using a ten seconds timeout [*Figure 2*]. I chose ten seconds for no other reason that it contains the best of both worlds. If the OWIN framework needs to look up and access a ApplicationUser object somewhere else in its pipeline, it is likely less than ten seconds from the initialization of the ASP.NET page.

If the object is older than that, then we must make sure to refresh the information. Otherwise, if an operation updates the database with new roles or passwords, the user would be forced to

```
Public Class ApplicationUserStore
    Implements IUserStore(Of ApplicationUser)
    Implements IUserPasswordStore(Of ApplicationUser)


    ...

    Public Function GetPasswordHashAsync(user As ApplicationUser) As Task(Of String)
Implements IUserPasswordStore(Of ApplicationUser, String).GetPasswordHashAsync
        Return Task.FromResult(Of String)(user.PasswordHash)
    End Function

    Public Function SetPasswordHashAsync(user As ApplicationUser, passwordHash
As String) As Task Implements IUserPasswordStore(Of ApplicationUser, String).
SetPasswordHashAsync
        user.PasswordHash = passwordHash
        Return Task.FromResult(0)
    End Function

    Public Function HasPasswordAsync(user As ApplicationUser) As Task(Of Boolean)
Implements IUserPasswordStore(Of ApplicationUser, String).HasPasswordAsync
        Return Task.FromResult(Of Boolean)(Not String.IsNullOrEmpty(user.
PasswordHash))
    End Function

    ...

End Class
```

Figure. 3

```
Imports Microsoft.AspNet.Identity

''' <summary>
''' This Class is used to decide the type of password received from the database
''' and generates a has value to compare it.
''' </summary>
Public Class ApplicationPasswordHasher
    Implements IPasswordHasher

    Public Function HashPassword(password As String) As String Implements
IPasswordHasher.HashPassword
        ' Return Clear Text as the Hash.  Not as secure, but needed when sending
password to
        ' program.
        Return password
    End Function

    Public Function VerifyHashedPassword(hashedPassword As String, providedPassword As
String) As PasswordVerificationResult Implements IPasswordHasher.VerifyHashedPassword
        ' No hash was done, so check clear text.
        If hashedPassword = providedPassword Then
            Return PasswordVerificationResult.Success
        Else
            Return PasswordVerificationResult.Failed
        End If
    End Function
End Class
```

Figure. 4

wait on the web server to decide if the data is old or not.

Interface IUserPasswordStore

Since we normally need to authenticate the user with a password, it is important to implement the IUserPassword-Store interface [*Figure 3*]. One of the gotchas with OWIN is that it uses Microsoft's default password hashing system. While this is good because it does not keep the passwords as clear text, it doesn't help us when we need to reset a password outside of the OWIN framework.

Sometimes you need to keep the password stored as plain text, or in a two-way hash system, in order to use the same password functions already built-in to your LOB (Line Of Business) system. In order to do this, you will need to create another class, out-

side the ApplicationUserStore class, to handle keeping passwords in clear text [*Figure 4*].

This class will be hooked up to the ApplicationUserStore when we implement the ApplicationUserManager.

Conclusion

These are the only interfaces that you really need to implement in order to make the OWIN Security framework functional. There are many more that I will go into in later articles that I think are also important. These are the minimum that are required to get something to work. **IS**

NATHAN RECTOR
President
International Spectrum
nathan@intl-spectrum.com

Date

BY CHARLES BAROUCH

In the opening to the TV show, Six Million Dollar Man, they promised to rebuild him – better, faster, and stronger. This occasional series, the Six Million Dollar _____ will offer suggestions on how to elevate mvBASIC commands by making them better, faster, and stronger. We have the technology.

Getting data into an mvBASIC program requires a READ or an INPUT. While I have seen many, many attempts at creating a wrapper around the INPUT command, the diversity of data we push through often makes the wrapper into a programming language unto itself, using dozens of parameters and conditionals to attempt a one-size-fits-all solution. Instead of trying to be *everything* to *everything*, our six million dollar DATE will be an input wrapper focused on getting date-related input right. Sometimes limiting scope is the best answer.

Dates take on many forms: most operating systems have a time-date stamp based on an epoch. MultiValue has a epoch based on date alone. Europeans write dates as day/month/year while Americans write month/day/year. People who care about sorting external dates use year/month/day. Then there's Julian format, longhand (1 June 1981), and shorter longhand (28 Sept 1985). Reasonably, our wrapper needs to understand all of them (plus

```
001 subroutine DATER(RESULT, RULES, DEFAULT)
002 * by Charles Barouch (Results@HDWP.com)
003 * as seen in International Spectrum Magazine
004 input ANS
005 convert '/' to '-' in ANS
006 if ANS = "" then ANS = DEFAULT
006 begin case
007    case ANS = matches '"T"-1N0N' and INDEX(RULES<1>,'T',1) = 0
008       * Example: T-1 (yesterday)
009       IDT = DATE() - oconv(ANS,'MCN')
010    case ANS = matches '"T"+1N0N' and INDEX(RULES<1>,'T',1) = 0
011       * Example: T+1 (tomorrow)
012       IDT = DATE() + oconv(ANS,'MCN')
013    case ANS matches '1N0N"-"1N0N"-"1N0N' and (RULES<3> = "A" or RULES<3> = "")
014       * American format (mm-dd-yy or mm-dd-yyyy)
015       IDT = OCONV(ANS,'D') ;* Assuming You are set to American format
016    case ANS matches '1N0N"-"1N0N"-"1N0N' and RULES<3> = "E"
017       * European format (dd-mm-yy or dd-mm-yyyy)
018       TEMP = oconv(ANS,'G1-1'):'-':oconv(ANS,'G0-1'):'-':oconv(ANS,'G1-1')
019       IDT = OCONV(TEMP,'D') ;* Assuming You are set to American format
020    case 1
021       IDT = ANS
022 end case
023 *
024 begin case
025    case RULES<2> = "I"
026       RESULT = IDT
027    case RULES<2>[1,1] = "D"
028       RESULT = oconv(IDT,RULES<2>)
029    case 1
030       RESULT = IDT
031 end case
032 return
```

Figure 1

two bonus cases to be discussed later) and should be able to convert them any-to-any for use [*Figure 1*].

RULES<1> has the allow/disallow for input formats, null means allow. RULES<2> has the desired resulting format. RULES<3> is either "A"merican, "E"uropean, or "J"ulian.

About that first bonus case: Date math and date shorthand are among the most common things needed by data entry. T+7 is much easier than making the user look up the date seven days from now. EOM is much easier than forcing them to memorize *30 days hath September, April, June, and November.* Personally, I like being able to type W-1 for last week or Sun+1 for a week from Sunday.

I've only put in a fraction of the code here because the rest becomes easy once you have the examples. To create week math (W-7 style logic) just copy lines seven through twelve and change "T" for "W" and multiply the oconv result by seven.

To create end-of-month (EOM style logic) just take today's date, keep the month and year and sub out the day for 28,29,30, or 31 as appropriate. To create day-of-the-week (SUN+1 style) divide the internal date by seven to get the SUN date. If MON was requested, get the SUN date and add one. SAT is SUN plus six. If you prefer SUN, MON, TUE... only being future, you can use the formula outlined and add seven if the result is less than today.

About that second bonus case: We often need to distinguish between skipped and intentionally left out. When I worked in rush courier, we had two special 'dates' called International and Unknown. Placing an "I" in a date field meant that there needs to be a date but it can't be determined until business hours at the international destination. "U" meant that there needs to be a date but it can't be filled in yet for any reason other than the one which "I" denotes.

Having a generalized date routine that can be plugged into any program makes adding or restricting features like this easy. Did I miss a format you use? You can add it to your version of the six million dollar date. **IS**

Got your own wrapper for a MultiValue BASIC command or TCL verb? Send us your Six Million Dollar make-overs so we can share them with the community. Email editor@intl-spectrum.com.
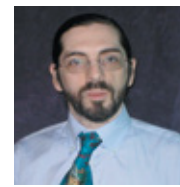
CHARLES BAROUCH is the CTO of HDWP, Inc. and the Publisher at HDWPbooks. You can read his writing in International Spectrum, Theme-Thology, Novo Pulp, Pax Solaria, PerehelionSF, and the Interrogative series, which begins with Tiago and the Masterless.

More About Dates

I've actually had to ask "What does a day mean?" For some companies, the day ends at closing time, not at midnight. Mom-and-pop shops might close their day whenever mom leaves because she won't be updating the financials again until tomorrow.

Oh, and if you think a day has twenty-four hours, consider that some have twenty-five due to DST (Daylight's Saving Time), and some have twenty-three, also due to DST. Of course, when to add or subtract that hour is locational. Australia and the U.S. have DST reversed from one another. And, inside the borders of Arizona for example, they never have DST. Well, except on Navajo land, where they do. Florida has a plan underway to make DST permanent (as in, never "fall back" to Standard Time).

Weeks and months are similarly complex. While the SEC (Security Exchange Commission) has rules against closing a month early or holding it open late, privately held companies can elect do do those things. So, assuming a month ends on the last day is not a sure assumption.

What about years? For most businesses, a year is made of full weeks, which means either a few days in December roll into the next year or a few days in January are owed to the previous year. And, for the historically minded, one year is missing eleven days. The day after September 2nd, 1752 is September 14th, 1752. At least it was in England.

You can read more about that here: http://mentalfloss.com/article/51370/why-our-calendars-skipped-11-days-1752

Revelation Software announces the release of OpenInsight Development Suite 10.0

Revelation Software today announced the availability of the latest release of OpenInsight Development Suite (OI) 10. OI 10 provides the following new and/or improved components:

• New IDE - OpenInsight 10 includes a brand new IDE, visually similar to modern application development tools, yet still designed for the MultiValue developer. Each of the product's designers, for forms, tables, reports and all others are now contained into a single, cohesive workspace.

• Arev64 – The next generation of our Advanced Revelation emulator, utilizing the full power and capabilities of modern 64-bit architectures.

• User and Data Security – With increased concerns about security and privacy, including upcoming GDPR requirements, a number of enhancements and changes have been made to strengthen data security.

• Management Console – The OpenInsight Management Console is a browser-based database management tool for system administrators.

• Git Integration – OpenInsight's repository based environment is integrated to the Git source code management system.

• Universal Driver 5.1 – 64-bit and VSS capability.

"This release of OpenInsight is revolutionary", said Mike Ruane, President/CEO Revelation Software, "We've rewritten nearly every bit of code in the product, from the interface with the user, to the reporting tools, and the speed of the indexing routines."

Also included in OI 10 are a number of enhancements for the MultiValue Base Filing System (MVBFS), Cloud Based Filing System (CBFS), NetOI/RevDotNet, OIPI.Net/OIPI, SQL Connector, Editor ++/Basic+, OECGI4 and CTO.

"The improvements to our O4W robustness is a huge breakthrough", said Robert Catalano, Director of Sales, Revelation Software. "Our new WYSISYG Form Designer makes creating responsive browser-based applications much easier, and much more intuitive."

OpenInsight Development Suite 10 is available from Revelation Software, or through their network of resellers. See Revelation.com for details. ∎

Synergetic Data Systems release CirrusPrint 2.0.13

CirrusPrint 2.0.13 has been released. This is a performance release that has been tested under heavy loads to ensure it will handle the high throughput requirements some of our users have reported, without overtaxing cpu and network resources. Both the server and system client have been improved, so Synergetic recommend all users update their server, then their system clients. ∎

Synergetic Data Systems release UnForm 9.0.31

UnForm 9.0.31 has been released. This update includes both bug fixes and enhancements. The enhancements include a new -asyncrpq option to allow direct writing to a local server's rpq directory, better archive tab management from the main portal page, and substitution tag support in email log files. ∎

SA Police Super Works with MBS to Update their System

S A Police Super have been using MultiValue in one form or another since 1992 to help administer our Superannuation Scheme. The original legacy software was developed by a Melbourne company and has since undergone monumental changes to bring it in line with the boundless legislative and taxation changes which relentlessly rain upon us every year. Not to mention all the user changes to make the system more effective and user friendly. As such, we have spent a lot of time and effort (not to mention money) to have what I would call a mature system. It does what it does and it does it fairly well given its legacy constraints.

We're now in the 21st century and it's time to upgrade the look and feel of the application, incorporating some of the latest technologies and industry standards, to ensure it is up-to-date and cutting edge. We have so much invested in our system; it's tailored to our needs and fits our business perfectly in terms of functionality, it just requires a more modern interface. There's only one of me however, and I'm spread pretty thinly, so what to do?

> *We have so much invested in our system; it's tailored to our needs and fits our business perfectly in terms of functionality, it just requires a more modern interface.*

With the above in mind I turned to our good friends at MBS to talk about moving into the 21st century using our existing D3 database. We needed something that would allow us to use existing business rules built into our system and provide a modern look and feel GUI interface to provide more meaningful displays than the old legacy 80x25 green screens.

MVS Toolkit to the rescue!

The folks at MBS showed us the MVS Toolkit which allows us to communicate with our D3 database via web services thus opening the door to a browser-based system using the existing business rules subroutines) and of course the ability to provide more meaningful "screens", i.e. web pages with way more detail than ever possible on an 80x25 green screen display.

There are a whole bunch of new technologies involved which need to be learned: HTML, CSS, jQuery, Ajax, and MVSToolkit. Best practices need to be researched, developed and implemented. It's not an "easy" road to travel, but it's sure interesting and exciting melding these modern technologies together with D3.

I'm by no stretch of the imagination an expert in any of these technologies, but I have programmed in FlashBASIC for decades and dabbled in C#, VB and HTML for a few years thus enabling me to understand and pick up these new technologies.

With the eager help of the skilled MBS Team I've been able to start putting together the beginnings of a new intranet web-based system which will take us into the 21st century and beyond. Yes, it's only 1 page, but it's the foundation upon which the new system will be built. **IS**

UniVerse and Python - it's here!

BY BRIAN LEACH

Some months back I wrote in Spectrum about Rocket's plans for introducing Python into UniVerse and UniData. Well, the wait is over because the release of UniVerse 11.3.1 — according to some at Rocket the biggest release they have ever done — is here.

So first off — a big 'well done' for getting it out the door.

I know I'm often critical of the product — I carp because I care — so it's great have the chance to give praise where it's due. 11.3 is a milestone release and one that promises interesting times ahead.

Whilst there are other changes, it's the introduction of Python that stands out. Not merely as a client language but as a fully-fledged server language sitting alongside UniBasic. This is a huge step forward not just for Rocket but for MultiValue in general.

Why Python Matters

We have not seen a new language adopted server side — well, not since Dick Pick added mvBasic all those years ago. Some brave souls may have embraced RPL (whose history predates Basic) or ALL in the old days. We have

> *Those of us who work with UniBasic... appreciate the clarity and intelligibility it can bring to a business process.*

had 4GLs and domain specific languages galore — including my own, of course, in the shape of the mvTest, mvScript and mvStudio products.

We have seen massive improvements to the core mvBASIC language, including extension libraries and syntactic improvements like the object orientation and JSON objects on QM. There have been link-ups like the way jBase can import C functions directly into its cross-compiled variant. And of course the language itself is no more BASIC than VB.NET — we've just been stuck with that name and the negative connotations that surround it. Given how eagerly the vendors have always gone out of their way to rename everything has also increased confusion (PROC to ProVerb, ACCESS to AQL, ENGLISH, INFORM, RetrieVe, UniQuery or jQL) it's amazing that in all that

time they haven't renamed the one thing that really needed it!

But a fundamental addition of a new language is a different ball game.

Those of us who work with UniBasic, especially in its UniVerse dialect and with knowledge of other languages, appreciate the clarity and intelligibility it can bring to a business process. Much easier to follow through a well written UniVerse routine than navigate the spaghetti mess of several dozen java classes (why does java never use one class where it could possibly spawn twenty?). Most business processes are essentially linear, and that suits a clear and simple procedural language.

Why then, after all these years, do we need a new language? The answer is simple — recruitment.

Why Python Really Matters

Just last week I was in a meeting with a client whose critical UniVerse application is under threat from senior management . It's a depressingly familiar story — the system must go, not because it fails to meet the needs of the users but because it is an alien technology in their business, not understood by the consultants they increasingly

use to outsource and because the have a problem finding and retaining skilled UniVerse developers. The fact that it has, like so many MultiValue systems, been starved of resources for years whilst keeping the business running on a shoestring with only one dedicated chap looking after it, has eluded them. As has the prospect of training up new developers to understand it.

So today we are drafting our response, pointing out all the usual stuff. No, UniVerse is not a legacy technology and is not older than the 'modern' SQL Server you're proposing to replace it (though it may look it, you do realize that "new" system is just Sybase with a pretty front end - you've heard of NoSQL haven't you?) and no, it's not going to disappear next year because you've never heard of the sup-

plier and yes, there are places you can go if it ever did, and yes, the application is old because you haven't funded it responsibly but it will work with a shiny new .NET front end. And we agree, per-user licencing is expensive and archaic in today's world but those licences have already been paid for.

Oh and by the way, your expectations for re-coding it all on SQL are wildly optimistic.

We've seen this all before.

So the technical arguments we know.

But the risk from under-staffing is a real one and ever present. Yes, we understand that your standard recruitment agencies don't have a clue what MultiValue is and so they can't find you new people, but you should know what there are specialist agencies who

probably can. But it's still seems to be so much easier, in business minds, to spend a few million rewriting a system in "mainstream technology" than a few thousand training up new developers to understand this one.

But now, there is a new and powerful argument.

Today you can hire Python programmers off the street and, with the knowledge of a simple API and guidance from senior people who do understand UniVerse and the application, put them to work writing UniVerse server code. It really is that simple.

From a perception point of view, this is massive.

'Why would I want to learn BASIC, that's a hobby language (thanks, Microsoft) not for professionals. It's a

dead end. If I learn it will I ever get another job?'

Python does not have that legacy. It is used everywhere — from embedded applications on boards such as the Raspberry Pi to large scale websites. It is easy to learn but not frowned upon. And it has a huge and enthusiastic following.

Getting to Grips

When Rocket first announced Python support, my initial reaction was the same as I hear from other people since — why not JavaScript, isn't that the language of the moment? But Python has integrated well and who knows, perhaps having made this leap, (Rocket are you listening) JavaScript may be an option in future. Or perhaps the hip young things wouldn't be seen using JavaScript on a server (except Node.js, which has inexplicably made itself cool).

If you've not used Python before it is weird and will trip you up, but it is easy to learn and there is a huge wealth of materials out there for both beginners and experienced programmers looking to add it to their repertoire.

It is structured, and whilst I appreciate good UniBasic code, it is probably harder to write bad code in Python than it is in UniBasic. At the least, anything written in it will be a fresh start, so no excuses.

Python routines are written in regular files and run using the RUNPY command:

```
RUNPY filename itemname
```

There is no catalog equivalent, though that would be nice to see, as would the ability to call a Python function directly from UniObjects. Right now

you would have to shim that through a UniBasic subroutine to make the call. UniBasic has been extended to integrate directly with Python.

And nice to see, Rocket has introduced some videos to explain these new features.

The most important thing to note is that this is not a bolted-on client language. The Python integration, whilst presented through UniObjects-like semantics, is in the same memory space. This is important as it means the performance is roughly on a par with UniBasic for such operations as file i/o. The performance tests I have run so far indicate that whilst there is an overhead — as one would expect — it is well within the tolerable range to make this a suitable language for writing server routines. The exact numbers will of course vary with platform specifics, but expect around 25% on a file operation compared to UniBasic. That's actually pretty good going.

To put this in context, just from my laptop:

Writing, reading and deleting 10 x 10,000 items took UniBasic 2.9 seconds, Python 3.8 seconds.

Just for fun, doing the same in PROC took 6 seconds.

I'll be blogging more about this, and the other 11.3 features, over the coming days but for now, grab hold of 11.3 and have a play. **IS**

BRIAN LEACH is a MultiValue developer, consultant, trainer and author, and a board member of the Rocket Software Users Group. You probably knew that. Find him at http://www.brianleach.co.uk.
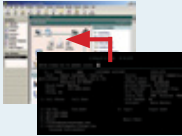
MARKETPLACE

ACCOUNTING

Natec Systems
www.natecsystems.com | nrector@natecsystems.com

COMPLIANCE

SJ+ Systems Associates
www.sjplus.com | sjoslyn@sjplus.com

CONSULTING

Drexel Management Service
www.drexelmgt.com | dconboy@drexelmgt.com

Execu-Sys, LTD
www.eslny.com | mh@eslny.com

HDWP
www.HDWP.com | results@HDWP.com

Modern MultiValue, LLC
www.ModernMultiValue.com | info@ModernMultiValue.com

PICK Programmers Shop
www.pickprogram.com | brian@pickprogram.com

Precision Solutions
www.precisonline.com | Kevin@PrecisOnline.com

DATABASE

Ladybridge Systems Ltd
www.ladybridge.com | sales@Ladybridge.com

REPORTING

Brian Leach Consulting, LTD
www.brianleach.co.uk | brian@brianleach.co.uk

TERMINAL EMULATOR

Zumasys
http://www.zumasys.com/products/accuterm/

LETTERS TO THE EDITOR

Have an opinion on an article: Agree, disagree, or enhancement to an article from a previous issue? International Spectrum and our authors are interested in hearing from you!

E-mail: editor@intl-spectrum.com

WANT TO SEE A SPECIFIC TOPIC?

International Spectrum is looking for writers, feedback, and topic ideas. We all have specific topics and issues that we need answers to find solutions for. Send us an E-mail with topics you would like to have covered in the magazine or on the website.

E-mail: nathan@intl-spectrum.com

WANT TO WRITE?

Expand your professional credentials, and provide us with an article.

Give us a rough and ugly outline, and we will help you refine it, proof it, and make it press ready. Or you can give us something polished, proofed, and press ready to publish.

Share your thoughts and expertise with over 10,000 fellow MultiValue developers and users.

E-mail: editor@intl-spectrum.com

NEED A MENTOR?

Mentors give developers the ability to ask industry experts for direction, code examples, and/or just ask them to see if something makes sense. Sometimes, all you need is a resource or example to start or complete a project.

Check with us to see who is available for mentoring, and how you can take advantage of it to save your business or company money.

E-mail: nathan@intl-spectrum.com

WANT TO BE A MENTOR?

We have many retired or semi-retired professionals out there that would love to share their knowledge of MultiValue development. If you are one of them, please contact us to see what mentoring is all about.

E-mail: nathan@intl-spectrum.com

Same Difference

BY CHARLES BAROUCH

Ask most successful business owners and you'll find that they attribute part of their success to doing things differently. They will tell you that they offer a special view on service, a unique sales proposition, or significant value-add to their process; something that sets them apart.

Our experience as consumers tells us a different story. While we all have our view on certain companies being exceptional — Microsoft, Apple, and Google have enthusiastic consumers for example — we usually see most of the businesses we deal with, personally and professionally, as close to interchangeable. There are slight preferences in many cases but we don't see huge differences.

Of course, everyone see categories of business as different. Few people would argue that your accounting firm and your pet groomer are interchangeable. For the purposes of this article, however, *they are*.

CSP

Most businesses are based on CSP - Cost Sharing Propositions. Hiring a full-time accountant (or becoming an accountant) is too expensive. Sharing

> *...we all want the people who pay us to be able to keep paying us.*

one accountant among two dozen clients provides that accountant with a living without laying the expense on any one client. My younger daughter, Dani, is part of a group which does dog walking and house sitting. Hiring a house sitter for only the days you will be away is CSP. My sister is in catering, where she creates and orchestrates large, elegant parties. Hiring her for the weddings and other notable events which occur over your entire life makes more sense than having her on payroll during the years between your two daughters' weddings.

While industries are different, and businesses within industries are different from each other, CSP is the point of commonality that most of them share. The practical question is: Why does do we care?

Atoms Matter

In developing software, as a consultant or employee, we can take the most specific approach: The Acme Corporation needs software for people who make the left-hand gloves on Tuesdays between seven and noon. Ideally, while accounting for the specifics, we also want to lift our heads up and try to see farther. Making software that helps people make any sort of glove, on any of the seven possible days — morning, noon, or night — will still let us target Acme's need. Additionally, as their needs change, our software will be more adaptable.

This is perspective. Looking up from the specifics and seeing the wider potential for our applications is how we move from writing custom software to writing verticals (software tailored to an industry) or writing horizontals (software tailored toward a functional set of activities, like inventory or customer service).

CSP is the next logical step: Moving from a vertical model or horizontal model to a more universal model. The core argument for CSP is that while gold and lead are different, they are the same because they are made of atoms.

For application software houses, focusing on the problem of *gloves* instead of *left-handed Tuesday gloves* is inherent in the approach. If they are particularly attentive, they will understand

the points of commonality between glove manufacture and other clothing manufacture. This is the CSP path, go from specific, to general, to hyper-general.

Growth and Death

While the Supreme Court may have indicated that corporations are people, I disagree. However, they do share the attributes of growth and death. This is where CSP comes in, as a help with the former and a hindrance to the latter.

I'll make the broad assumption that we all want the people who pay us to be able to keep paying us. Whether we are in business for ourselves and those people are the customers, or we are employed and those people are the bosses, their success helps us. What defines success? In a capitalist endeavor like business it is measured in growth and money.

Larger, more diverse, audiences for your product or service increase the ability to expand. A product that only has one customer is harder to make profitable. The more customers, the more opportunities to make a profit. CSP is a viewpoint that helps you generalize what you do.

Those who know me know that I used to be a senior executive in a half billion dollar clothing company. The division I was initially employed by made belts, ties, and suspenders. Our divisional software was working so well that the pitch was made to move the entire company to our platform. While only half the company ended up moving, it was still a daunting challenge.

We could have looked at the problem and — rightly — said that belts have one size, ties have no sizes, so our software should not be used for shirts which have two sizes (collar **and** sleeve
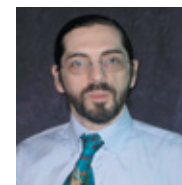
measurements). We could have pointed out how ties and belts are singular products (one unit is one unit) but suits are mutlipart products (one unit is a roll-up bill of materials of other finished parts, like pants and jackets). We didn't. We took a CSP view and found the common points. By isolating the differences, we solved the problems.

I won't tell you I was the hero of this story because I wasn't. I contributed, I cheered others on, and I helped where I could. It worked because the entire team accepted the idea that it could and should work.

The Resume

I recently helped someone with their resume. Their understanding of the differences between their current job and the job they wanted got in their way. Being less well informed, I suffered no such problem. Instead, I looked at the job description, looked at their resume, and drew parallels.

As technical people, we live in the details, the minutiae, the weeds. Sometimes taking the ten-thousand-foot view helps us see more clearly. Her resume, our view of the company we work for, both of them can benefit from an occasional step back. You may discover a brand new place where we fit in. **IS**

CHARLES BAROUCH is the CTO of HDWP, Inc. and the Publisher at HDWPbooks. You can read his writing in International Spectrum, Theme-Thology, Novo Pulp, Pax Solaria, PerehelionSF, and the Interrogative series, which begins with Tiago and the Masterless.