$7.00 U.S.

INTERNATIONAL Spectrum®

THE MULTIVALUE TECHNOLOGY MAGAZINE | MAY/JUNE 2015

International Spectrum 2015 Conference Wrap-up

Also In This Issue

- Call(Overhead)
- Lumpy! Exploring What Makes a Database a MultiValue Database
- Using Inner Recursion
- Clif Notes — New Blood Part 3

INTERNATIONAL Spectrum®
THE MULTIVALUE TECHNOLOGY MAGAZINE

FEATURES | MAY/JUNE 2015

DEPARTMENTS

From the Inside

I need authors for articles. I need content providers. Are you interested?

I've asked many people to write articles over the past few years and have been rewarded with some very interesting material for the magazine. Some of the best articles are the ones provided by the end users. Most of these authors are techies like you, not professional writers.

I understand that writing can seem daunting. You may wonder where to find the time to do it. Well, we are here to help.

The most common problems that new authors face:

- "I don't know what to write about."
- "My company places restrictions on revealing propriety code."
- "I'm not a writer and don't know how to create a finished article."
- "The things I do aren't that special. Everyone know how to do them."

Issue: What to Write About

I have to come up with a topic for the "From the Inside" every issue. I can understand how daunting this can be. While it seems like this is the biggest challenge, in reality it is the easiest for you to address.

The answer: Call me or e-mail me and ask.

I have a whole list of topics for articles. Most of time, however, I want to hear about the solutions you have developed for your business. You already have a story or two to tell.

On the other hand, if you are reading this and would like to see a specific article in the Spectrum Magazine, please feel free to contact me and let me know. I can add it to my growing list of articles, webinars, and conference topics.

Issue: Not a Writer

Well, truth be told, neither am I. My spelling is horrible. My grammar makes Chuck and Syd pull their hair out. And in nearly every article I write, they have to clarify at least one paragraph because it's not quite clear what I'm trying to say. [Editors' note: Sob.]

My suggestion is: Don't try to be a writer. Be yourself. Write the article as if you are talking to someone or making notes on how to do something. We can take that rough material and make it into a finished, polished product.

Your submission doesn't have to be a word document. It can be something typed out in e-mail. Take an hour for lunch and use your smart phone to write it.

So, if you are worried about your writing.... relax. Let us address the fine points of style and grammar. Trust me, the International Spectrum Staff is very good at doing this.

Issue: Proprietary Code

This is a serious issue. We don't want you to reveal anything confidential. You don't have to provide your company's code in the article. Just provide the problem you were solving and an outline of the solution you developed.

Many times the article is about a concept or suggested way of doing something. In those cases you can strip the examples down to a few lines of code to show the problem,

INTERNATIONAL
Spectrum
MAY/JUNE 2015

NATHAN RECTOR
President

CHARLES BAROUCH
Editor

SYDNEY BAROUCH
Editor

TRACEY RECTOR
Layout

MULTIVALUE

Learn more about the MultiValue Symbol and see what MulitValue Technologies and MultiValue Communities exist to help you support and manage your business and systems. To find out more visit
http://www.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.

International Spectrum is published six (6) times per year at the subscription price of $40.00 U.S. in the U.S.A.; $45.00 U.S. in Canada and Mexico; $50.00 U.S. for other countries. Single copy rates are $7.00 U.S. in the U.S.A. and Canada, and $9.00 U.S. in all other countries. International Spectrum is published by International Specturm, Inc., 3691 E. 102nd Ct., Thornton, CO 80229; Tel: 720/259-1356; Fax: 603/250-0664 E-Mail: request@intl-spectrum.com. Copyright 2015 International Spectrum, Inc. All rights reserved. Reproduction in whole or in part, without written permission, is prohibited.

PRINTED IN USA

NEWS RELEASES/UNSOLICITED ARTICLES
International Spectrum is eager to print your submissions of up-to- the-minute news and feature stories complementary to the MultiValue marketplace. Black and white or color photographs and diagrams are welcome. Although there is no guarantee a submitted article will be published, every article will be considered. Please send your press releases, articles, and queries to: editor@intl-spectrum.com. International Spectrum retains all reprint rights.

without giving away anything your company owns.

One of authors here at International Spectrum had this same problem. He collaborated with his management, and worked out a way to write the article, covering the topic without using any of the company code. Most of what he supplied was information provided on public domain websites, and only a line or two of specific code.

No company processes or secrets were given way.

Issue: Everyone Already Knows How to...

You'd be surprised.

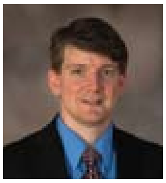
I see users, admins, and developers every year at the Spectrum Conference – next one will be on April 11th-14th, 2016 – that are new to MultiValue databases. They need to understand many of the core features that we in the community take for granted.

Here is an example from 2009: When someone asked, "How can I print my MultiValue data as a PDF document?" We could have said, "Well everyone knows how to do that." Instead, we published a series of articles covering that topic. <http://intl-spectrum.com/mag/MAYJUN.2009/default.aspx> <http://intl-spectrum.com/mag/JULAUG.2009/default.aspx> <http://intl-spectrum.com/mag/SEPOCT.2009/default.aspx> Even people who already knew how can often benefit from a 'deep dive' into the subject.

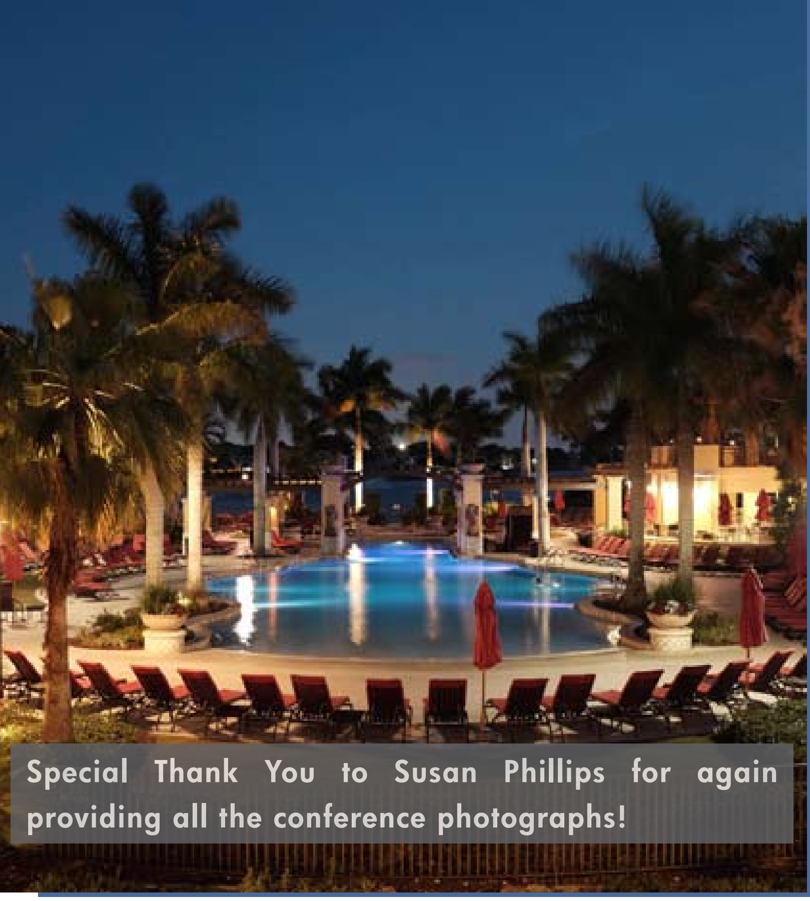
Just because some topic or technique seems like common knowledge, that doesn't mean there aren't people who don't know how to do it.

I need authors for articles. I need content providers. I want to hear how you solved your business or technical problems.

Email me: nathan@intl-spectrum.com



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



Special Thank You to Susan Phillips for again providing all the conference photographs!

INTERNATIONAL Spectrum®

2015 MultiValue Conference Recap

As usual, the International Spectrum Conference provided MultiValue developers, end users, and administrators a place to meet, talk, and network with other professionals in the MultiValue Community. This year it all happened in West Palm Beach Gardens, Florida.

Just like last year, we reformulated the offerings to focus more on start-to-finish tracks. Attendees looking for education in MultiValue database technologies and integration were able to walk away with over 20 hours of professional development training. And that was just what they got inside the classrooms. We supplemented those with how-to and design-heavy sessions.

There were many interesting conversations this year. Three topics in particular:

- Millennials
- Agile Testing
- Green Screen To GUI

Nathan Rector, President of International Spectrum, talked about Millennials and how they are changing the business software landscape. This focus was underscored by the many sessions that included tips, topics, and examples that helped attendees adapt their business software to the new workforce demands. One of the core issues is an expectation of flexibility and personalization in business software comparable to what we now find in leisure software. Many of our sessions offered solutions.

Of course, not all changes are driven by workplace turnover. There is a constant push, across many verticals, to replace green screen applications as part of a general refresh of stale business applications. This topic was addressed in the classrooms and on the exhibition floor.

We didn't just talk about changes in how we write software. Testing, specifically Agile Testing, was an important topic this year. This is driven by the habits of .NET and Java developers. They've championed this approach in their application development. Our sessions on Agile Testing talked about the tools and design requirements needed to create an Agile Testing framework that works for MultiValue.

Other sessions covered topics that ran the gamut from the basics of HTML – and best ways to control styling and layout – to efficient ways for pulling your data out of your MultiValue database in real-time (for example, converting MultiValue data into JSON).

Mobile and tablet development continued to be a hot topic. In particular, the differences between a WebApp, NativeApp, and a HybridApp, and how to use them to effectively present business information.

As always, the International Spectrum Conference included sessions on creating desktop applications with .NET. These sessions showed MultiValue developers how to leverage .NET GUI and event processing to create powerful UIs and tools. There were also sessions teaching .NET developers how to access MultiValue databases effectively using event driven programming.

We also had several valuable management-based topics: Project Estimation, MultiValue Source Control with GIT, Talking with SQL Programmers, Big Data in MultiValue Databases, Disaster Recovery Planning, and SQL vs. MultiValue.

With many businesses looking closely at whether to buy new hardware or virtualize their existing systems, the sessions on virtualization answered questions on speed, storage, and design considerations. They also offered checklists for when-and-if you plan to virtualize into a private or public cloud.

Finally, MultiValue Fundamentals hit the spot for many of the newer attendees and novice MultiValue developers. For the last three years there has been an increase in attendees that are new to MultiValue databases. This year's sessions included BASIC programming, dictionaries, and MultiValue file design. We provided attendees with a grounding in the environment, in a format that allowed our more senior attendees a forum for mentoring and answering questions that were not covered specifically in the sessions.

On Tuesday and Wednesday the attendees were able to talk with our sponsors about tools and products that would help their businesses. We kicked that off with our *Vendor Speed Dating* which is always a blast.

For those that do not know what *Vendor Speed Dating* is, each sponsor gets five brief minutes to talk to the attendees before the *Sneak Peek Party*. It gives the attendees an idea of what each exhibitor is offering. If they like the speed date, they can enjoy a more leisurely visit on the show floor. The event is designed to be fun and very informal. We had one sponsor sing a song to the attendees, another gave a dramatic presentation — complete with a cape — and third read an excerpt from his upcoming science fiction book.

In addition to the classes, the hallway conversations, the speed dating, the exhibition floor, and the impromptu demonstrations in the lobby, attendees gathered in various places to network with each other over lunches and dinners. Of course, Thursday's activities ended with our traditional *Closing Conference Party*. Everyone had one last time to talk, laugh, and network with each other while they enjoyed the food and drinks.

Bringing the experience home, most of the attendees took advantage of our post-conference downloads and session videos as well. Access to these materials is just one of the perks of attending the live conference. It is often helpful to be able to revisit popular sessions through our on-demand recordings, with the downloads provided by the presenters and sponsors.

Now that this year's conference is over, it is time to start planning for the next one. If you have any topics you would like to see at the Spectrum Conference, please let me know. I am always looking for new ideas and new presenters. Contact me at nathan@intl-spectrum.com.

International Spectrum 2016 will be at The Wigwam in Litchfield Park (Phoenix), Arizona on April 11th – 14th, 2016. Don't miss it. **IS**

Thanks to our 2015 Sponsors!

Ashwood Computer Company

BBWC

DesignBais International

Drexel Management Service

Eagle Rock Information Systems

Entrinsik

HDWP

Ladybridge Systems Ltd

Onsystex

Pick Cloud, Inc.

PICK Programmers Shop

Rocket Software

SJ+ Systems Associates

Syngergetic Data

Zumasys, Inc.

Call(Overhead)

BY KEVIN POWICK

The Challenge

For decades, MV BASIC has featured the `CALL()` statement, allowing for external subroutines invoked by mainline programs. Typically, one does this to share common code/processing/algorithms, thus upholding the development principle of DRY (*Don't Repeat Yourself*). However, such features often come with a performance penalty. The question is, *"How much of a performance penalty?"*.

Recently, while reading through the D3 Reference Manual for version 9.2.1, I noticed some performance tips mentioned in the *Compiling Programs* subsection of the chapter on *BASIC*. In particular, this statement:

To obtain the best possible compile-time and run-time performance, Rocket recommends breaking up applications into small modules. Pick BASIC removes the traditional run-time overhead of large numbers of calls and is able to create more efficient code when modules are smaller

The part of that stood out to me was, *"Pick BASIC removes the traditional run-time overhead of large numbers of calls…"*. Oh, really? That sounded like something worth testing. So, I did.

The Method

I didn't have a lot of free time, so I knew from the start that my approach would be simple. All I wanted to do was quickly discover how much of a performance penalty there might be using a subroutine `CALL()` vs. in-line code. Since I was doing the research, I decided to also test if making local subroutine calls via `GOSUB` had any significant impact on performance.

Not wanting to consider file I/O, my program only did a simple mathematical calculation within a tight `FOR/NEXT` loop.

```
FOR THIS.CALC = 1 TO MAX.
CALCS
   RESULT = ((THIS.
CALC*THIS.CALC) + THIS.
CALC) /  THIS.CALC
NEXT THIS.CALC
```

To keep it somewhat flexible for others to test their own platforms and scenarios, I included several control options — the user can control how many times each test runs, the number of iterations of the `FOR/NEXT` calculation loop, and the `SYSTEM()` code used to get the CPU time in milliseconds for their particular MV implementation.

Since my main platform is D3, I tested both Flashed and non-Flashed versions of the programs.

The Results

The tables below show, on average, how much longer the test program took to run through its calculations when using `CALL()` vs. in-line code.

Flashed Code

Iterations	x In-Line Time
25,000	1.59
50,000	1.50
75,000	1.48
100,000	1.47
250,000	1.49
500,000	1.50
750,000	1.21
1,000,000	1.10

Non-Flashed Code

Iterations	x In-Line Time
25,000	4.00
50,000	3.88
75,000	4.03
100,000	3.95
250,000	1.52
500,000	1.36
750,000	1.32
1,000,000	1.32

It should not be surprising that there is a performance penalty when using `CALL()` vs. in-line code. What *is* surprising is how the penalty changed as iterations increased. I expected the penalty to be somewhat consistent, or possibly exhibit a slight linear increase as iterations increased, but the results suggest otherwise.

Conclusions

"Pick BASIC removes the traditional run-time overhead of large numbers of calls…".

That's what Rocket asserts and I set out to discover. I knew from the beginning that using `CALL()` would impose a penalty, but after testing, the real question became, *"Does it matter?"*.

I guess that depends on the nature of your operation and what you're trying to achieve. What's not shown in this article are the execution times of the tests. Without considering those times, you're only getting part of the entire picture.

For Example, if it takes 46 seconds to run 1,000,000 iterations using inline code, it will then take 4 seconds longer (46 x 1.10 = 50.6 seconds) to use an external call. Is that significant enough to justify dropping the convenience of `CALL()` to in-line the code and save those seconds? For me, the answer is no.

So, in defense of Rocket's assertion about `CALL()`, I would say that I agree with them. For the vast majority of processing, the convenience of using `CALL()` to keep your code modular and DRY, is going to far outweigh the performance gains by in-lining.

Two Final Notes.

1. Flash-compile your code. It's significantly faster than non-Flashed code.
2. The difference between in-line calculations and executing them via `GOSUB` is not significant enough to even consider dispensing with the convenience of `GOSUB`. (60ms differential on 1,000,000 iterations).

The Code

Download the code from the FOS-S4MV repository at Bitbucket. **IS**

GET CONNECTED.

Knowledge and Education for the MultiValue Professional.

About Our Professional Membership

We are all busy in our day-to-day work and staying up-to-date with the current MultiValue technologies can be difficult.

Professional Memberships provide you access to knowledge, solutions, information, and code that you won't find in other locations.

Professional Membership Includes:

- Magazine in Electronic and Print Formats
- Newsletter
- On-Demand Videos
- Live Webinars
- Discounted Conference Rates
- Research papers
- Case Studies
- Source Code

INTERNATIONAL Spectrum®

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

AccuTerm Mobile v1.0.6 Available

AccuTerm Mobile allows users to access their MultiValue systems from any mobile device running Apple iOS or Android. The app is optimized for connecting to MultiValue databases but can connect to any host server running Telnet or SSH.

AccuTerm Mobile provides industry leading terminal emulation for ADDS, Wyse & VT terminals, and offers integrated mobile features such as browser, email, and phone functionality for use with legacy applications. Users can take advantage of both SSH and telnet connectivity, allowing you to connect to any MultiValue, Linux, Unix or AIX host system on the go, including jBase, OpenQM, D3, UniVerse, and UniData.

New Features in version 1.0.6 include:

- Added setting to select terminal screen font size, including an option for auto-sized font that scales to fill the screen width with the appropriate number of columns.
- Added support for AccuTerm mouse click reporting (this works the same as the desktop version of AccuTerm, except there is no "hand" cursor (because there is no mouse). Only single tap (click) is supported, and it is reported as the "left button." See the AccuTerm 7 help, AccuTerm 7 Reference chapter, AccuTerm Programming section for details.

Bug Fixes:

- Fixed SSH bug which did not recognize SSH server configured to support both SSH v1 & v2.
- Fixed problem when using Bluetooth keyboard on iPad, where lower half of terminal screen was unused.
- Fixed Android 4.1, 4.2 & 4.3 problem where soft keyboard would automatically close when the Enter key was pressed.
- Fixed problem programming function keys using AccuTerm private ESC sequence.
- Fixed problem with underline & blinking effects not always working.
- Minor Wyse & ADDS emulation bug fixes.
- Fixed minor VT visual attribute problem where wrong colors were used if the attribute was reset to normal in a separate ESC sequence from the attribute being set.
- Changed terminal type string sent to host for Viewpoint A2 Enhanced emulation from 'wyse-vp' to 'viewpoint' because many older Linux versions do not recognize 'wyse-vp'. Changed QM terminal type string for Viewpoint A2 Enhanced from 'vpa2-at' to 'adds4000-at' to match the string used in the desktop version.
- Removed Android file access permission requirement.

If you have purchased AccuTerm Mobile, this update is free. ■

Rocket Software Introduces the Rocket Discover Business Intelligence Solution

Rocket Software is pleased to announce the launch of Rocket Discover, a self-service, intuitive data preparation and discovery solution which gives business managers and executives the speed and confidence to make key decisions. Rocket Discover allows users to easily access, manipulate, prepare and visualize data to expose insights and help them make critical business decisions that will drive growth and outpace the competition.

"Companies facing critical business issues know that they need to act fast and act smart, but they often don't have the right information at their fingertips to investigate and quickly resolve the problem," says Bob Potter, Senior Vice President and General Manager of Business Information and Analytics at Rocket Software. "While there are many self-service BI solutions on the market, we continue to hear about critical functionality that these solutions lack: self-service data preparation, access to the mainframe, and scalable performance. With this in mind, we built Rocket Discover to be a world-class visual data discovery tool for business users that addressed these gaps."

Now business users can easily access and enrich data from multiple sources – ranging from mainframes to databases to the cloud. With self-service data preparation and discovery in one solution they can get business insight within minutes, not days or weeks. For optimal performance Rocket Discover was engineered with a data-retrieval architecture that combines in-memory performance with live query access, a hybrid approach that can handle today's high volume thresholds.

Today's business professionals know their data, their business challenges, and the need to quickly connect the two to address these challenges. Rocket Discover's emphasis on intuitive design for self-service data preparation and discovery makes this possible. Users can easily interact within a rendering canvas to create visual data process flows and dashboards through simple drag-and-drop movements. And when it's time to collaborate, team members simply drag a dashboard into the chat window for immediate dialogue on strategy and opportunities. Rocket Software has specialized in database connectivity, preparation and solving customer problems for over 25 years, which is why Rocket Discover is differentiated with:

- Connectivity to disparate data that includes IBM z Systems and IBM Cognos TM1
- Self-service data preparation and data discovery in one solution
- Data Lineage for governance
- High Performance data access for rapid decision making

Jan Baesjou of ABM Systems, a Rocket Strategic Partner, says that Rocket Discover is redefining BI. "We have used Rocket's BI products for over 20 years, and Rocket Discover is the perfect addition to our business planning arsenal. There are plenty of BI tools out there, but most of them focus mainly on dashboarding. Rocket Discover has capitalized on their data preparation IP in which they've excelled at for many years. Having simplified data preparation within a BI solution is a key feature and it is a win for our business users."

"We have been using Rocket Discover for several months, and are impressed with the potential that the application presents," says Rachel Byrom from Oxford City Council. "I come from the IT side of the business, so I'm excited to get Rocket Discover into the hands of as many of our business users as possible. Rocket Discover will allow staff to directly access data and create dashboards that are contextual to their needs, when and how they need it. This will ultimately reduce the number of requests to my team for reports and dashboards, allowing us to focus more on development projects and system improvements."

Subsequent releases of Rocket Discover will continue to capitalize on Rocket Software core competencies by providing additional features in the areas of data connectivity, maximizing the value of the mainframe, increased levels of intelligent guidance during data preparation, and enhanced collaboration.

About Rocket Software

Rocket Software is a global software development firm that builds enterprise products and delivers enterprise solutions in the following segments: Business Information and Analytics; Storage, Networks, and Compliance; Application Development, Integration, and Modernization; and Database Servers and Tools. Rocket is engaged in business and technology partnerships with IBM, EMC, Fujitsu, Hitachi Data Systems, Avaya, Epicor, Infor, and many others. The company is headquartered in Waltham, Massachusetts, USA. ■

Roll Your Own (Highly Experimental) MultiValue Database Part 1

By Charles Barouch

The Vendors' Lot in Life

In this series of articles, we are going to build a brand new flavor of MultiValue database. A brand new storage/retrieval engine which understands MultiValue. To be clear: *It is not our intention to create a full-featured product*. This is just to give you a feel for the basic functions that our vendors make work — across multiple hardware platforms and operating systems — despite constant changes in technology. Because they do their jobs well, it can be easy to forget how much effort goes into creating and maintaining a product of this complexity.

I've nicknamed this new engine 'Lumpy' because the data isn't flat. Lumpy *will* be a viable database for learning the core concepts of MultiValue. It could be useful in the classroom. You will have the source code; you can share it and extend it. However, I wouldn't run my business on it. Not even a lemonade stand.

Some of the things it *won't* be:

- encrypted,
- secure,
- scalable to hundreds of users,
- scalable to Big Data sizes,
- "brained" (more on that below),
- capable of supporting an optimized file system,
- …. and you get the idea.

The brain of a MultiValue database handles locking, licenses, and other centralized functions. Lumpy is brainless. None of those things will be addressed in this article series. You can kinda/sorta do all of that in your programs which use Lumpy. It isn't the best solution, but we aren't here to create a seamless, perfect system.

We will have a way to do optimistic and pessimistic locking in a 'brainless' mode. Encryption, security, identity, integrity, these things need a brain — and a custom file system.

This is barebones. Lumpy exists to give a deep dive into the thought process needed to create a database engine. We will produce a working model of how CRUD (Create, Read, Update, and Delete), AMS (Attribute, Multivalue, Subvalue), and very basic database management work.

The articles in this series will include PHP code, which performs all of these functions. You will be able to use PHP to write programs which incorporate Lumpy's functionality. I have no intention of writing a BASIC compiler. Well, maybe as a retirement project.

If we get enough feedback (send it to editor@intl-spectrum.com) I will also post additional examples and tutorials. If you ask nicely, I might develop versions for other programming languages. Remember, since you have the source code, I'm not the only one who could port Lumpy.

What Makes a Database a MultiValue Database?

If we are going to call Lumpy a MultiValue database, it might be handy to establish a definition. Here's what I came up with after a discussion with Kevin King:

1. Non-typed fields. We won't enforce typecasting in the engine. PHP, our programming language, is very lightly typed.

2. Fields do not have a predefined length.

3. The option of architecting data into a 3NF format (good SQL is 3NF). The key here is the logical grouping of data within tables.

4. Ability to store data in a real-world format, i.e. store every part of the invoice in a single record structure instead of being forced to parse it for the sake of SQL's rigidity.

5. Inferred from #4: nested data.

6. Have a query processor to help filter and arrogate data.

7. Forced uniqueness of IDs.

8. MD/VOC - Lumpy will:

    1. use a PHP include to define the actions (Verbs) and other syntax, and

    2. use a system manifest, which acts as a pointer to each of the dict/data areas.

Creating a Database

For clarity, we have chosen a simple manifest to manage the structure of our database. While this is much more primitive than what the vendors do, it is functionally similar. The only structural commands we will implement are three create commands and their corresponding delete commands. Note: The delete will not do any 'Are You Sure?' steps. This is barebones.

`CREATE-DATABASE` and `DELETE-DATABASE`

Syntax examples:

```
include_once('./verbs.php');
Lumpy->command('create-database spectrum');
Lumpy->command('delete-database spectrum');
```

Manifest Example:

```
[Lumpy]
database=spectrum
```

The code for verbs.php can be found at hdwp.com/lumpy/

We will add the `delete-database` code in the next article.

CREATE-ACCOUNT and DELETE-ACCOUNT

Syntax examples:

```
include_once('./verbs.php');
Lumpy->command('logto-database spectrum');
Lumpy->command('create-account magazine');
Lumpy->command('delete-account magazine');
Lumpy->command('create-account
              → distribution');
Lumpy->command('create-account magazine');
Lumpy->command('create-account
conference');
```

Manifest Example:

```
[Lumpy]
database=spectrum
account=spectrum,distribution
account=spectrum,magazine
account=spectrum,conference
```

The code for verbs.php can be found at hdwp.com/lumpy/

We will add the `DELETE-ACCOUNT` code in the next article.

Creating a Table

For clarity, and to make the speed reasonable, we are going to store data as JSON records. For more about JSON, read the sidebar.

The only commands we will implement are `CREATE-TABLE` and `DELETE-TABLE`. Note: The delete will not do any 'Are You Sure?' steps. Our create will let us make tables that are single JSON records or use a simple hashing scheme to split a file across multiple JSON records.

Manifest Example:

```
[Lumpy]
database=spectrum
account=spectrum,distribution
table=spectrum,distribution,customer
table=spectrum,distribution,customer,
            customer
table=spectrum,distribution,orderheader
table=spectrum,distribution,orderheader,
            order,1,101
```

Note: The hashed tables add two values. In this case: 1 tells us which hashing algorithm, 101 tells us the modulo.

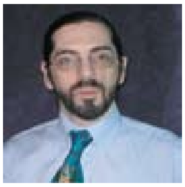
```
table=spectrum,distribution,orderdetail
hashedtable=spectrum,distribution,
            orderdetail,order,2,301
table=spectrum,distribution,shipto
hashedtable=spectrum,distribution,shipto,
            shipto,2,11
table=spectrum,distribution,ordersummary
hashedtable=spectrum,distribution,
            ordersummary,yr2015,1,101
hashedtable=spectrum,distribution,
            ordersummary,yr2014,1,101
hashedtable=spectrum,distribution,
            ordersummary,yr2013,1,101
```

Note: The table allows for multiple data levels associated within one table structure.

```
table=spectrum,distribution,ordersummary,
            yr2014
table=spectrum,distribution,ordersummary,
            yr2015
```

Next Up

In part two, we will show you how to CRUD (Create, Read, Update, and Delete) up your database. **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, PerehelionSF, and the Interrogative series, which begins with Tiago and the Masterless.

JSON

JSON is a data format originally designed for use with Javascript. Like XML, JSON uses embedded tags, but the ones in JSON are less repetitive, yielding a more compact format. While not as efficient as pure MultiValue, PHP and Javascript both have excellent parsers for JSON. That's why we are using it as Lumpy's storage format.

Here's the same data in XML, JSON, and MultiValue:

```
XML (409 characters)
<set>
  <record>
     <id>2015-05</id>
     <issue>Apr/May 2015</issue>
     <format>pdf</format>
     <format>ebook</format>
     <content>
        <article>Are You a Technology
Bigot</article>
        <by>Kevin King</by>
     </content>
        <article>Business Tech</article>
        <by>Charles Barouch</by>
     </content>
        <article>From the Inside</
article>
        <by>Nathan Rector</by>
     </content>
   <content>
        <article>New Blood</article>
        <by>Clif Oliver</by>
     </content>
  </record>
</set>

JSON (281 characters)
{"2015-05":[
    {“issue”:”Apr/May 2015”,
      “format”:[“pdf”,”ebook”],
     “content”:[
    {"article":"Are You a Technology
Bigot", "by":"Kevin King"},
    {"article":"Business Tech",
"by":"Charles Barouch"},
    {"article":"From the Inside",
"by":"Nathan Rector"}
    {"article":"New Blood", "by":"Clif
Oliver"}
    ]}
]}

MultiValue (146 characters)
000 2015-05
001 Apr/May 2015
002 pdf]ebook
003 Are You a Technology Bigot]Business
Tech]From the Inside]New Blood
004 Kevin King]Charles Barouch]Nathan
Rector]Clif Oliver
```

Using Inner Recursion

BY KEVIN KING

Unbalanced Trees

One of the more common complex structures found in a variety of applications is that of the unbalanced tree. Because we are MultiValue professionals, we tend to take complexity in stride, but recursion can still be a stumbling point.

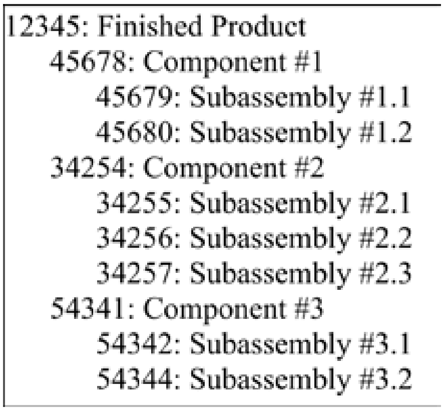
An unbalanced tree structure could be used, for example, to represent a bill of materials. The tree would illustrate the relationship between a parent part number and any number of child part numbers, and the relationship between each of those child part numbers and their children, ad infinitum. Such a structure might appear in Figure 1.

In MultiValue, this can be represented very simply by having items in a file, with a multi-valued list of components stored in one of the record attributes in each record. The structure would appear somewhat like Figure 2.

In this file, attribute 20 nominates the components (shown above as Component #1 through Component #3) which are used in the construction or definition of this item. Each of these items would then in turn have components defining the individual subassemblies.

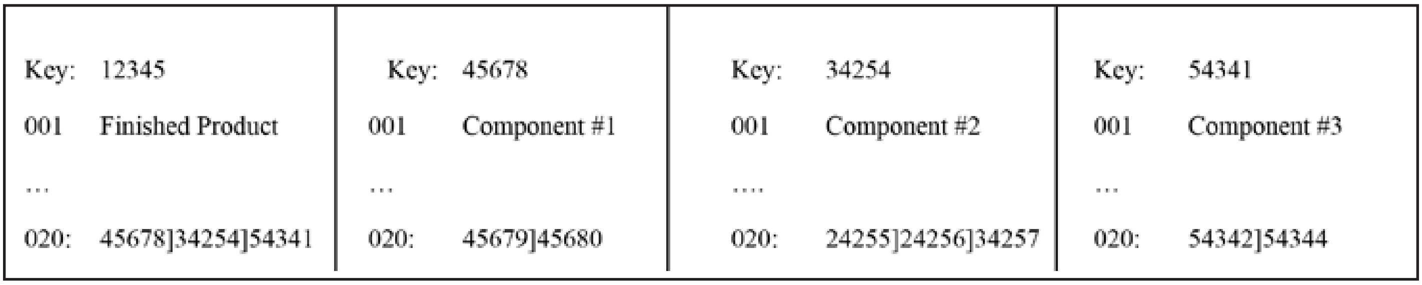
Obviously, by implementing this structure using MultiValue technology, any number of levels can be defined for the structure without going outside the boundary of a single file. As long as a component has components of its own, this structure can implicitly link the records together without indices or a multidimensional inverted list. Each parent keeps track of their children, allowing the whole structure to remain intact. (This, incidentally, applies just as easily to a supermarket as it does programming.)

Storing the structure is only half the battle, however. Eventually someone will want to retrieve the information on a report or inquiry. If the information is stored in a tree structure, we are obligated to follow the tree when printing the report. While this is a reasonable expectation, I'm often amazed at the heinous array of code fragments



```
12345: Finished Product
    45678: Component #1
        45679: Subassembly #1.1
        45680: Subassembly #1.2
    34254: Component #2
        34255: Subassembly #2.1
        34256: Subassembly #2.2
        34257: Subassembly #2.3
    54341: Component #3
        54342: Subassembly #3.1
        54344: Subassembly #3.2
```

Fig. 1



```
Key:  12345                 Key:  45678           Key:  34254                 Key:  54341
001   Finished Product      001   Component #1    001   Component #2          001   Component #3
...                         ...                   ....                        ...
020:  45678]34254]54341     020:  45679]45680     020:  24255]24256]34257     020:  54342]54344
```

Fig. 2

littering the path to this sort of objective.

Traditional Recursion

Probably the most obvious technique for tree traversal is simple recursion. With this technique, a routine is written for handling a single item. Once the item has been processed, the routine extracts any descendant data and calls *itself* for each descendant. Such a routine might appear something like Figure 3.

In this example, the ID of the primary inventory record is passed as the parameter. Once that item has been processed, the children are extracted, and each child ID is passed to the PROCESS.ITEM routine. This will cause the entire tree to be traversed, as the routine will repeatedly call itself for all descendant data.

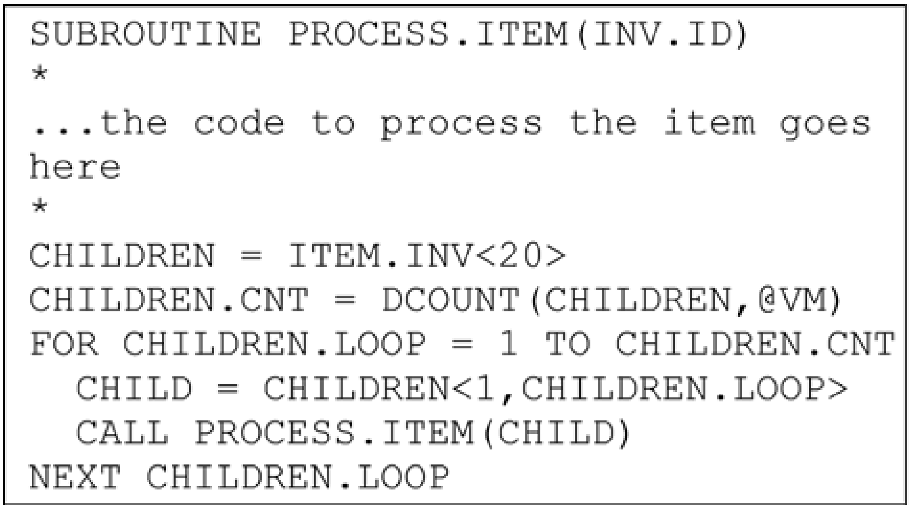
Having said this, and having shown it to be so simple, it should be noted that this particular technique is not generally a good idea for MultiValue systems. Instead of taking the space to fully explain the reasons behind every one of the problems, I'll point out two major ones: (A) Inordinate quantities of memory are required simply to support the recursion, and (B) it's dog slow. Apologies to the dogs.

Inner Recursion

To overcome these problems, a technique called inner recursion can be used. Like traditional recursion, inner recursion is a technique by which a routine calls itself for processing descendant data. Unlike the traditional method, however, this technique does not have hefty resource requirements or speed problems. In fact, this technique uses very little additional memory to support the recursion, and traverses a tree remarkably quickly.

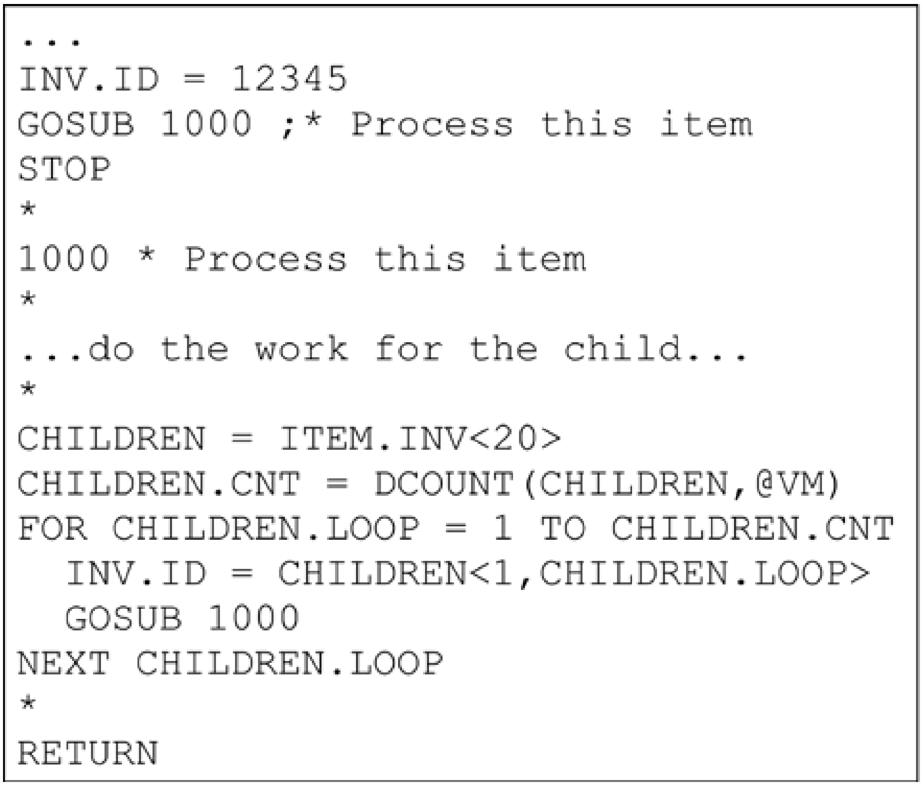
The trick – if I can call it that – is that the routine uses GOSUB to call itself instead of CALL. The astute among us will instantly see that there is a serious problem with this approach: Internal subroutines do not have private variables. In other words, if a subroutine calls itself, the called instance does not have its own copy of the variables, and any variables changed by that instance will remain changed after that instance RETURNs to its caller. Consider the example in Figure 4.

This code fragment will work fine if there's only one level of components. However, if there's an additional level of components – as shown in the subassembly example earlier (Figure 2) – the CHILDREN, CHILDREN.LOOP, and CHILDREN.CNT variables for the first level of components will be changed when the routine calls itself, thus destroying their original contents



```
SUBROUTINE PROCESS.ITEM(INV.ID)
*
...the code to process the item goes
here
*
CHILDREN = ITEM.INV<20>
CHILDREN.CNT = DCOUNT(CHILDREN,@VM)
FOR CHILDREN.LOOP = 1 TO CHILDREN.CNT
  CHILD = CHILDREN<1,CHILDREN.LOOP>
  CALL PROCESS.ITEM(CHILD)
NEXT CHILDREN.LOOP
```

Fig. 3



```
...
INV.ID = 12345
GOSUB 1000 ;* Process this item
STOP
*
1000 * Process this item
*
...do the work for the child...
*
CHILDREN = ITEM.INV<20>
CHILDREN.CNT = DCOUNT(CHILDREN,@VM)
FOR CHILDREN.LOOP = 1 TO CHILDREN.CNT
  INV.ID = CHILDREN<1,CHILDREN.LOOP>
  GOSUB 1000
NEXT CHILDREN.LOOP
*
RETURN
```

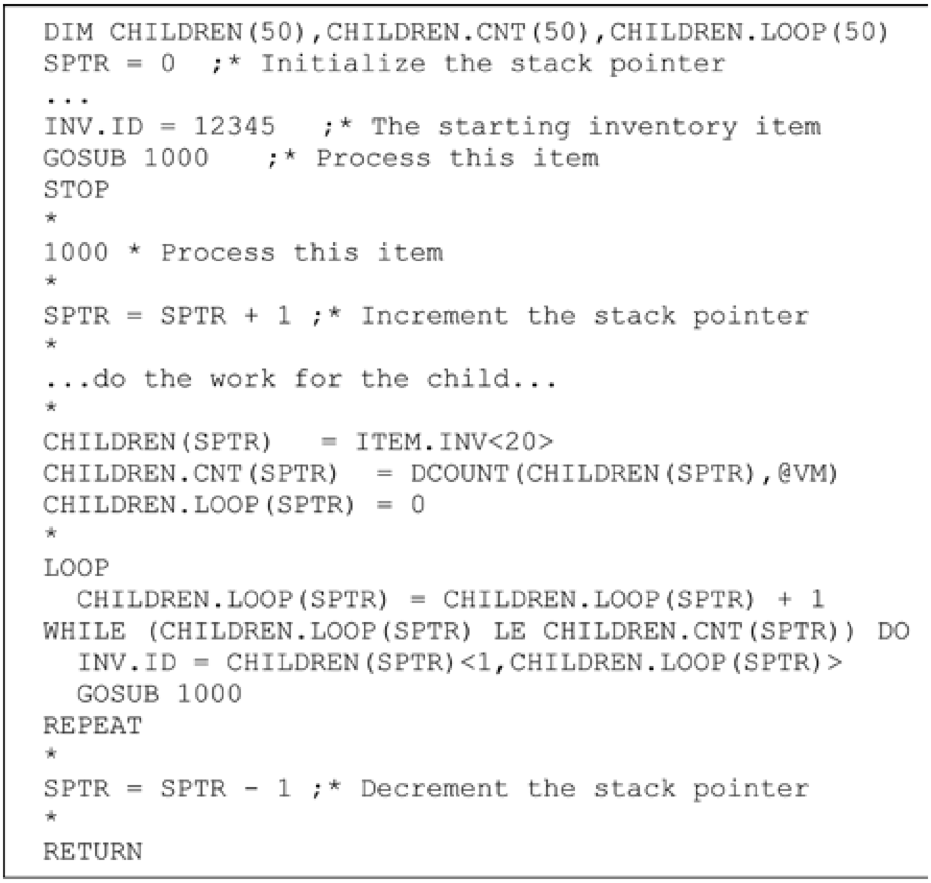
Fig. 4

and causing the end result to be incorrect.

Fortunately, this problem is easily remedied. We simply need to identify those variables whose values are unique to each given instance, and convert them to arrays as in Figure 5.

Note the DIM for the dimensioned arrays at the top of this fragment. If the maximum size of the structure is known, and the structure is not too big, dimensioned arrays are always the best performance option. If the maximum size of the structure is unknown, or the structure could grow to an inordinate size, dynamic arrays could be used instead, though notably with a performance penalty.

For each iteration of the 1000 subroutine, note how the stack pointer, SPTR, is incremented at the top and decremented at the bottom. This is used to maintain autonomy for each instance of the CHILDREN, CHILDREN.CNT and CHILDREN.LOOP variables. The first time through, SPTR is incremented to 1. As 1000 calls itself, SPTR is incremented to 2, 3, etc. As each level is completed, SPTR is decremented to its previous value, so when all is said and done, SPTR is returned to the calling routine with its original value of zero.



```
DIM CHILDREN(50),CHILDREN.CNT(50),CHILDREN.LOOP(50)
SPTR = 0  ;* Initialize the stack pointer
...
INV.ID = 12345   ;* The starting inventory item
GOSUB 1000    ;* Process this item
STOP
*
1000 * Process this item
*
SPTR = SPTR + 1 ;* Increment the stack pointer
*
...do the work for the child...
*
CHILDREN(SPTR)    = ITEM.INV<20>
CHILDREN.CNT(SPTR)   = DCOUNT(CHILDREN(SPTR),@VM)
CHILDREN.LOOP(SPTR)  = 0
*
LOOP
  CHILDREN.LOOP(SPTR) = CHILDREN.LOOP(SPTR) + 1
WHILE (CHILDREN.LOOP(SPTR) LE CHILDREN.CNT(SPTR)) DO
  INV.ID = CHILDREN(SPTR)<1,CHILDREN.LOOP(SPTR)>
  GOSUB 1000
REPEAT
*
SPTR = SPTR - 1 ;* Decrement the stack pointer
*
RETURN
```

Fig. 5

Also note that the FOR/NEXT loop from Figure #4 has been replaced by a LOOP/WHILE. While a FOR/NEXT may very well work, thanks to a bad syntax experience on an old ADDS Mentor, I've used LOOP/UNTIL or LOOP/WHILE ever since, and have never encountered any portability issues. And in this age of mergers and acquisitions, one should never discount the value of portability.

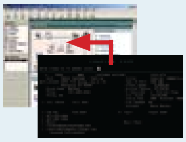
My point in writing this is simple: To those who have written heinous code to achieve tree traversal (you know who you are), hopefully this has impressed upon you that there is a different way. To those who need to do this kind of a thing, and just didn't know where to start, perhaps this gives you some food for thought. And to those for whom this is old hat, I just wanted to let y'all know I'm still alive and well. **IS**

KEVIN KING is the President and Chief Technologist with Precision Solutions, Inc., a leading technology solutions provider in Longmont, Colorado. He can be reached by email at Kevin@PrecisOnline.com or by voice at 303/651-7050.

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

J Squared Software, LLC
www.jsguaredsoftware.com
30+ years experience Business Applications &Support
MultiValue platforms - per diem or fixed price
Juddlove@jsguaredsoftware.com 203-325-0275

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

Entrinsik
www.entrinsik.com | sales@entrinsik.com

TERMINAL EMULATOR

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

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

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

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

Clif Notes

New Blood — Part 3

BY CLIF OLIVER

If it looks like a duck, walks like a duck, and quacks like a duck, it must be a duck. Right? If it looks like a dinosaur, waddles like a dinosaur, and grunts like a dinosaur, it must be a dinosaur. Stands to reason? If it looks like an obsolete programming language, is coded using obsolete tools, and is tested using archaic methods, MultiValue must be obsolete. Make sense?

I am always surprised, when I talk about the topic of uppercase-only coding, how angry and defensive people get. I really don't understand it. It would seem to me that if something as simple as changing the appearance of our coding style would help make us more acceptable to the Millennial developer cohort we're seeking to entice into our MultiValue ranks, that would be a no-brainer. But the resistance I get when I venture into this topic is, to me, astounding. It's all the way from, "Really? You think uppercase versus mixed case is important?" to, "THAT'S BULL!!" and they effectively stick their fingers in their ears.

Well, fine. I really don't care one way or the other what you think about uppercase-only code. Frankly, you don't have to give a whit about what *I* think about uppercase code.

Remember, the premise of this series is that MultiValue either is — or is in danger of — declining. In order to prevent this and grow the MultiValue space, we need new developers to both replace those who are leaving, and to provide an expanding pool of talent that will make MultiValue a viable business option for years to come. That means attracting developers from the Millennial generation.

Do Millennial developers care about *which* case code is written in? You bet they do. Don't take my word for it. Ask some of them. I did. In an International Spectrum magazine interview in 2011, I had an opportunity to ask a bright, young, promising developer who had just accepted a position with a MultiValue shop about his initial impressions <http://intl-spectrum.com/mag/MAYJUN.2011/default.aspx>. In his words:

Me: As a younger professional with a computer degree which is fairly recent (within the last five years), what was your reaction when you first walked in and saw everybody sitting here using green screens?

Him: Well, to be honest, at first, before I started learning anything about the language, I probably thought for a split second, what did I get myself into? It wasn't so much the green screens that worried me, but the everything being in ALL CAPS — which I found out since then, is not an actual requirement. That's more of a holdover from when it used to be a requirement. I know a lot of developers my age and probably younger now are more accustomed to using camelCase in our development. And IDE's are handy.

Did you get that?

"What did I get myself into?"

"Hold-over" (From the Old Days)

"My age and younger." (The developers we are supposedly interested in attracting.)

"camelCase" (C++, Java, C#, etc.)

"IDE" (Like Visual Studio, Eclipse, and others)

Here's an exercise for you. Go to the bookstore this weekend and browse the computer section. See how many of the books give their programming examples in all upper case.

Remember, first appearances are important. We are trying to sell these folks on coming into our environment. As many salespeople will tell you, you are selling feelings about your product and how it would feel using your product just as much (some might claim more) as the product's actual capabilities. You sell the sizzle, not the steak.

Want to put the nail in this coffin? Take them on a tour of the development area and let them see developers not just coding in uppercase, but doing so using the ED line editor. (I'm not making that up; I know several of them.)

If I can get this through to people, you would think that would pretty much be the end of the discussion, wouldn't you? But no. Now the arguments about the nature of reality start.

"Well, (Huff) It Shouldn't Matter!"

Excuse me? *Shouldn't* matter? I thought we just established that it *did* matter? To come back with the argument, "Well, it shouldn't," isn't valid. It has nothing to do with reality. It's just an opinion about the way the world ought to work, even though it doesn't.

I'm not going to argue with you on that. Who can? That's as valid as someone who, when told, "smoking cigarettes increases the chance of developing lung cancer" immediately replies with the brilliant counter-argument, "Uh, uh. I don't believe that!" and lights up. You simply cannot have a rational discussion with an irrational person. Don't try. Just move on.

The Users Don't Care; They Don't Look At Our Code. They Just Want It To Work.

True. But that's not the group we're discussing. We are talking about the people who will be looking at our code and are in that Millennial group we are trying to attract. Besides, you're right. They don't care. And they won't care – they won't care if it is written in mixed case, in MultiValue Basic, in SQL Server, Oracle, .NET, Java, Python, whatever.

The C-levels, VPs, Upper Managers, Etc. Don't Look At Our Code.

You are, in most cases, correct again. So where are these decision makers and check writers getting the idea that MultiValue is obsolete?

From their subordinates. Who also don't look at your code. So why are they saying we're obsolete? Where are they getting their information?

From their subordinates. And they get it from…

Follow the chain and you will eventually find a group of IT Millennials somewhere who have seen your code, your dictionary listings, etc. They see all uppercase and pronounce, "This is obsolete," and proclaim that up the chain to their supervisors and so forth. I've seen this several times. Somewhere, there is a group, maybe on the data warehouse project, maybe in the online commerce and web store team, who starts spreading the word that, "that old MultiValue is obsolete." And even if you don't have that in your organization, your CIO will hear it from their CIO colleagues at other companies.

Why Should I Have To Change The Way I Code? / I Can Code Faster In Uppercase Only. / I Don't See Any Reason…

Interesting what all of the objections in this category have in common. Think about that. In the meantime, let me share an experience I had with you, and see if you can see how I also got myself caught in this trap.

When I first started learning Python, I quickly grew very fond of the language. But I hated the style of coding they used. All lowercase with words separated by, for Pete's sake, underscore. Like this:

```
lines_left = lines_per_page - lines_printed
```

One of the things I like about camelCase is that I don't have to use another character to separate words, like a period that has a certain meaning to many having worked with various object-oriented languages. Underscore? Awkward. A shift AND a reach for the top row with the ring finger. That will slow me down.

Did you see what I just did there? I fell smack into the trap I tease other developers about when they say, "The shift key slows me down." (Tell that to typists who burn past you and me both going 75 WPM.)

Then, and I'm not making this up, I decided "I don't have to change the way I code just because a group of fanatics insists that their style manual, called PEP 8, says that's the Right Way to do it. And who crowned Guido the King, anyway?"

And I started using my beloved camelCase naming conventions and the Python compiler was still my friend. End of discussion. Until the next day…

I was looking at what I had written before and thought, "May not look like Python, but it works." Why should I care if it isn't "Pythonic?" So it doesn't make me one of the "Pythonistas." What difference does that make?

Oh. My. I suddenly realized that I was doing exactly what I'm saying we shouldn't do. *Major blush!*

I decided I needed to back down and re-think this. I'm a professional. I want to be viewed as such. If I want to be taken seriously by the existing professional Python crowd, I need to conform to those standards. I don't want "Pythonistas" looking at my code and laughing. It's a profession pride thing. (And as a consultant, it's a $ thing, too.) So I re-wrote that program and have been coding PEP 8 style ever since. It didn't take long for that ring finger to find the _ key as quickly as my little fingers find the shift key.

So I do understand these objections to changing the way most of us were taught to (and had to) code MultiValue. But it doesn't matter. If we keep doing what we've been doing, we'll keep getting what we've got — declining numbers, no respect from "mainstream" professionals, and no New Blood.

It's time for us to change and move along. It's a matter of survival, folks.

Of course, it doesn't do a bit of good for us to change and use a modern-looking coding style if there are people in the shop telling others in the department, "MultiValue doesn't do XML." "MultiValue can only exchange data by dropping CSV files in a directory." And other "Can't Do" claims.

We'll take that up in Part 4. **IS**

Informer™
Quick and Easy Business Intelligence
The BI Survey 14 ranks Entrinsik Informer #1 in "Ease of Use" and "Chosen as Standard" categories among dashboarding & ad-hoc reporting vendors.
Find out why at www.entrinsik.com
BI SURVEY 14
BARC