The Minimum You Need to Know

About Java on OpenVMS
Volume 1

By Roland Hughes

Logikal Solutions

ISBN 0-9770866-1-5

Acknowledgments

I would like to dedicate this book to you, the reader. You have chosen to work with the greatest operating system ever created even if it is no longer in vogue. You have endured the demeaning salaries and attitudes presented to you by what passes for management these days. You have stayed in the trenches and fought the good fight.

There is one person I would like to honor at this point. Not because he has anything at all to do with the subject matter, but he, like OpenVMS, managed to touch an awful lot of lives during his brief stay on this planet. That person would be Mr. Peter Jennings. We never met in this life, yet his voice found its way in on many an evening. Whenever tragedy struck somewhere in the world it was his newscast I found myself turning to, for not just the story, but the story behind the story. Some times you learned more from the side show spot than the actual newscast put on by other networks. In many ways I find myself identifying OpenVMS with those same broadcasts. It is the OS behind the big accomplishment. The story behind the story.

Source Code License

Any person owning a copy of this book may use the source code from this book and accompanying CD-ROM freely when developing software for their personal use, their companies use, or their client's use. Such persons may include the source code either modified or unmodified provided that the source delivered makes reference to the original author and is delivered as part of a fully functional application. It is expressly forbidden for anyone to post this software on any bulletin board system, internet Web site, or other electronic distribution medium without the express written consent of the author. It is also expressly forbidden to sell this source as part of a library or shareware distribution of source.

Users of the source code contained within this book and on the accompanying CD-ROM agree to hold harmless both the author and the publisher for any errors, omissions, losses, or other financial consequences which result from the use of said source. This software is provided "as is" with no warranty of any kind expressed or implied.

Table of Contents

Page left blank intentionally.

Introduction

I.1 Why Java?

That's a very good question. It's a question I've asked myself every time I've had to touch the language. If you grew up during the days of the languages we covered in the first book of this set, it is a question you will ask yourself over and over again. There is no good answer — none what-so-ever.

Considering I'm about to write an entire book on areas of the language not normally covered, that opening paragraph probably shocks you. Sadly, it's the truth. What is even more sad is that management makes its decisions based upon four-color glossies, and there are a lot of four-color glossies for Java products these days. The honest answer is "Because of the four-color glossies".

What inevitably happens in this market is management keeps trying to find "cheaper resources". This leads them to fish lower and lower in the skill pond. Many of you reading this are working in mainframe or midrange shops with an awful lot of PC hackers floating around. If you were paying attention, you watched it happen.

It started with someone from Mahogany Row reading a bathroom length article in the weekly trade press. There it spouted that if you weren't on the Web you wouldn't be in business for long. These articles all occurred prior to the Dot-Com flame out, which lead to the nickname DOT-BOMB.

A rash of static Web pages went up. These were followed by the ability to actually send email. Along the way, somebody said "Hey! Wouldn't it be nice if customers could actually place orders using this?" Coding up those kind of Web pages by hand was a lot of work. The skill and IQ level of college students had been "dumbed down" to the level of Visual Basic and other "visual" products. None of the current crop of "visual" tools worked well on the Web, but never fear, a rash of "visual" type tools using Java soon appeared. This allowed what is currently passing for college graduates to become employed creating Web sites and since they were recent graduates, they were cheaper than skilled professionals.

Once the Dot-Com flame out happened, Mahogany Row sat back and smiled. They were able to pick up a lot of Web and Java programmers for less money than they paid the box stackers in the warehouse. A mass proliferation of little Web applications sprang up all around your company. The wheel was re-invented at least a dozen times per quarter. Graphics got added to Web pages until the internal network was slow pulling the pages up. Nobody even thought about the poor Schmoe dialed in with a 14.4 modem, or worse, a 56K modem that only got a 14.4 connection from their ISP. You couldn't even walk in the machine room because everybody working on Web pages had to have their own server. Right around that time is when someone said "Hey! Wouldn't it be nice if customers could actually place orders using this?"

So, another rash of Java oriented commercial products hit the market. This time they were "middleware". Every vendor was hawking them and hawking N-tier systems design. At about this point, alert readers began to realize that Java is some of the most expensive "freeware" they could ever work with.

When you come up through the ranks working on OpenVMS, you cannot begin to understand why anyone wants an N-tier design. You've always had a cluster to work with. You've designed systems to use ACMS servers on every node in the cluster, consequently, you've never had to deal with load balancing because QTI did it for you. You've used RDB for your database because it was cluster aware. With DTM wrapping your transactions in its protective cloak, you've never had to worry about transaction recovery because all transactions were guaranteed to complete. With nodes of your cluster scattered across multiple states, you never even had to worry about fail over. As long as at least one node stayed up you were golden. Yes, the node would be gasping for breath being run so hard, but it would run. Let me put it to you in terms that will make sense: N-tier is what you do when you don't have a good platform to work with.

The concept of N-tier is pretty much "The Swarm Principle". If you mass together enough of an inferior force, you can overrun a superior force. Anyone who has ever watched TV has probably seen at least one commercial or movie where some hapless Schmoe is being attacked by bees, hornets, wasps etc. The principle works good from a military standpoint, but only if you are on the attack. From a defensive standpoint, once the superior force decides to attack, if you don't counter attack, you've lost before the first shot was fired.

Sadly, businesses who went looking to boost sales, discovered the failings of The Swarm Principle. They found out that if they had actually gotten all of the sales that tried to get through they wouldn't need to boost at all. Without an operating system that had both clustering and a distributed transaction manager built into its core, many sales were lost when these replicated nodes and services went down. They weren't stolen by competitors or identity thieves, the node which was supposed to process them grabbed the message, then promptly jumped off a cliff. When the customer didn't get a response within a reasonable amount of time, they logged into a different vendor's site to place their order.

Many of the companies which have made this discovery are beginning to move more and more of their Web back onto the OpenVMS cluster. Some are reconfiguring PC based Web servers to do nothing more than serve up pages and build messages for the OpenVMS cluster to process. Others are moving the Web server itself back onto OpenVMS to get a completely stable and robust platform for order processing and customer service.

To answer the question "Why Java?" is to understand why you, as a seasoned developer with other languages on this platform, are being asked to learn Java. The answer is two fold. Management wants to convert all of the core business systems to Java in order to get some use out of the Web developers they have on staff when they are not developing Web pages. While they know Java, they have expressed little to no interest in learning the applications of the core business systems. The second fold is that once the

Web servers move back onto the cluster, you, the core systems developer, are going to have to design and develop Web services from a systems, rather than a stand alone Web page perspective. The PC based Web page writers simply aren't capable of this.

I.2 Approach of This Book

You may have guessed from the opening section of this book, that it won't be like any other book you pick up on Java. Virtually every book on Java which I have either bought or read starts you off with a handful of stand alone classes, then immediately drops you into Web page development. In "Volume 1", we won't even cover Web page creation or development. At some point, I may write a "Volume 2" which covers that portion of Java from the OpenVMS perspective, but it is not currently in my plans.

We are going to cover the ugly side of Java which only gets a casual mention in most other books, JNI (Java Native Interface). As an application developer porting a core business system from one of the languages we previously covered to Java, you will face JNI on a regular basis. It is my hope that the source code provided both in this book and on the CD will jump start your application porting. You should be able to either use directly, or steal heavily from the source this book provides.

The native interface is a back door provided in the Java language to make it more usable. Using this back door typically requires more skill than most people using GUI development tools possess. You have to program in C or C++. Most Java developers wait for some serious chip-head to produce a new VM (Virtual Machine) which allows them to access the hardware and operating system features directly from Java. Others buy a middleware set of classes and saddle their company with a never ending support contract. My assumption in this book is that you don't have the luxury of waiting for a new OpenVMS friendly VM and don't want the never ending support contract which comes from yet another third party product, you need to start porting today. In fact, creating a VM which is OpenVMS specific is actually a violation of the Java license. HP could create a complete set of OpenVMS specific classes which gives Java developers complete access to the beauty of OpenVMS, but I wouldn't hold my breath waiting for that to happen.

Once we get through the obligatory basics of Java, we will jump straight into the JNI. Our first venture into the JNI will be accessing Run Time Library and System Service calls from Java. Once that is done we will write JNI code to access the RMS indexed files used in our Mega Zillionare application. After that we will once again employ JNI to access FMS. Once all of the steps are taken, we will develop the RMS indexed file version of our application in Java. Finally, we will move on to the same application using RDB. There won't be as many chapters in this book because we are covering only one language and moving at a much faster pace. I had to lay a lot of ground work in the first book. Now, I can make you jog, if not run.

I.3 Prerequisites for This Book

It is assumed that you have in your possession a copy of "The Minimum You Need to Know to Be an OpenVMS Application Developer" ISBN 0-9770866-0-7. (Hereafter referred to as "Book One".) It is also assumed that you have read that book and are at least somewhat comfortable with development on the OpenVMS platform. I will not be spending much, if any, time discussing the concepts covered in that book. You are expected to either know them, or look them up in that book.

You may wish to get your own Alpha machine and Hobbyist license. This will let you break things any way you want. While you could shop around on Ebay and hope against hope you bid on a machine which was correctly configured, I would highly recommend you contact Island Computers (www.islandco.com) to get a machine correctly configured. Then all you have to do is install your Hobbyist media and licenses. Many of you may be quite comfortable assembling PC's from scratch, but you have to be a little more selective with the hardware you get for an Alpha.

I.4 Obtaining and Installing Java

Hopefully, you are all corporate employees with systems managers keeping your OpenVMS machines properly configured. In a commercial environment software and service contracts lead to CD's showing up in the mail with the latest ECO's and software upgrades. I believe ECO stands for Engineering Change Order, but have used ECO for so long I cannot say for certain. This is the term used to describe all fixes and enhancements. When you wish to install the latest version of Java for OpenVMS you need to be at the current ECO level for it.

While it is true that Java does ship with OpenVMS, the version on my machine did not have the javah tool. This could have been a bone headed move on my part during the install of the products, or it simply might not have been included with the Hobbyist media. In either case, the version which was installed was 1.1.8-5 and the current version was 1.4.2.

Do not skip the step of downloading and applying all of the ECO's. Lots of people try to do this and it hurts them. I obtained my Java stuff from the following link:

http://h18012.www1.hp.com/java/download/ovms/1.4.2/sdk1.4.2_patches.html

While you could go to www.hp.com and look for Java stuff, I recommend you start off at www.openvms.org when researching Java for your machine. The site is a valuable resource for any OpenVMS developer, and its links are usually pretty current.

Be careful when you start to download ECO's and install them in the correct order. Some of them are 40+MEG in size (assuming you are running 7.3-1 which is the current Hobbyist version of OpenVMS). You may need a download manager or a very forgiving ISP. The other option is to contact HP directly and purchase the CD's containing ECO's

and Java for your machine. From a simplicity standpoint, that last option is the best, but it depends on the depth of your pockets for this adventure. The instructions for applying the ECO's are pretty straight forward. They appear on the Web site where you download the needed files.

Another thing you will want to download and install from HP is the Mozilla Web browser and email package. You will want the browser so you can open the Java documentation. Once Java is installed, the Java documentation is found in SYS$COMMON:[JAVA$1xx.DOCS]INDEX.HTML where xx = your version. Please note that you will need to enable DECWindows on your system console, or have an X-windows terminal emulator in order to run the browser. You cannot run the browser from a standard VT character cell interface.

If you wish to be an enterprising little geek, you can enable NFS services on your Alpha machine, obtain an NFS client for your Windows machine, and map the documentation directory into your local machine. This will let you open the HTML files from your PC browser. Telling you how to set that up is currently beyond the scope of this book. There are also very few "free" NFS drivers/clients available for the Windows XP platform at this time. If you are connecting to your OpenVMS machine from Linux, you may have an easier time of it.

Once you have downloaded either directly to your OpenVMS box or FTP'ed it from your PC you can run the installation process. Here is the session from my machine:

```
$ set def dkb200:[scratch]
$ run dkb200:[eco]DEC-AXPVMS-JAVA142-V0104-24P2-1.PCSI_SFX_AXPEXE
UnZipSFX 5.40 of 28 November 1998, by Info-ZIP (Zip-Bugs@lists.wku.edu).
  inflating: DEC-AXPVMS-JAVA142-V0104-24P2-1.PCSI
$ product install *

The following product has been selected:
    DEC AXPVMS JAVA142 V1.4-24P2           Layered Product

Do you want to continue? [YES]

Information has been saved to allow you to uninstall the following patches:

RECOVERY DATA SET 001 created 9-JUL-2005 17:13:10.69
   ------------------------------------ ------------------------------------
   PATCH                                APPLIED TO
   ------------------------------------ ------------------------------------
   DEC AXPVMS VMS731_PTHREAD V4.0       DEC AXPVMS VMS V7.3-1
   ------------------------------------ ------------------------------------

RECOVERY DATA SET 002 created 9-JUL-2005 17:12:05.81
   ------------------------------------ ------------------------------------
   PATCH                                APPLIED TO
   ------------------------------------ ------------------------------------
   DEC AXPVMS VMS731_DCL V5.0           DEC AXPVMS VMS V7.3-1
   ------------------------------------ ------------------------------------

RECOVERY DATA SET 003 created 9-JUL-2005 17:11:09.67
   ------------------------------------ ------------------------------------
   PATCH                                APPLIED TO
   ------------------------------------ ------------------------------------
   DEC AXPVMS VMS731_TDF V5.0           DEC AXPVMS VMS V7.3-1
   ------------------------------------ ------------------------------------
```

```
RECOVERY DATA SET 004 created 9-JUL-2005 17:09:47.54
   ------------------------------------ ------------------------------------
   PATCH                                APPLIED TO
   ------------------------------------ ------------------------------------
   DEC AXPVMS VMS731_ACRTL V4.0         DEC AXPVMS VMS V7.3-1
   ------------------------------------ ------------------------------------

* If you continue, recovery data for the patches listed above will be deleted.
* The deletion of recovery data does not affect the installation status of
* patches applied to products that are not participating in this operation.
* However, continuing with this operation prevents you from uninstalling
* these patches at a future time by use of the PRODUCT UNDO PATCH command.

Do you want to continue? [NO] yes

Configuration phase starting ...

You will be asked to choose options, if any, for each selected product and for
any products that may be installed to satisfy software dependency requirements.

DEC AXPVMS JAVA142 V1.4-24P2: JAVA for OpenVMS Alpha
    Copyright 2003 - 2005 Hewlett-Packard Development Company, L.P.

Do you want the defaults for all options? [YES]

Do you want to review the options? [NO]

Execution phase starting ...

The following product will be installed to destination:
    DEC AXPVMS JAVA142 V1.4-24P2          DISK$SYS:[VMS$COMMON.]

Portion done: 0%...10%...20%...30%...40%...50%...60%...70%...90%...100%

The following product has been installed:
    DEC AXPVMS JAVA142 V1.4-24P2          Layered Product

DEC AXPVMS JAVA142 V1.4-24P2: JAVA for OpenVMS Alpha

    Consult SYS$COMMON:[JAVA$142.DOCS] for the Release Notes for this product.


$DIR SYS$COMMON:[000000]

Directory SYS$COMMON:[000000]

JAVA$142.DIR;1             1/18        9-JUL-2005 17:22:26.42
JAVA.DIR;1                 1/18       17-NOV-2004 11:11:08.18
```

Each version of Java now has its own directory in SYS$COMMON. Long ago things were kind of scattered. You may wish to add the following line to your LOGIN.COM file or to SYLOGIN.COM if you own the system:

```
$ @sys$common:[java$142.com]java$142_setup
```

I.5 Major Pitfalls of Java on OpenVMS

I highly recommend that you poke around on www.openvms.org and find the downloads for ODS-5 prior to doing any development with Java. There is a procedure for converting your disk drives to ODS-5 (from ODS-2) listed with the download. Without trying to get too technical, using ODS-5 (which you would need if you installed MySQL when we covered it in our previous book) allows for mixed case file names and a more peaceful existence with opensource software.

Java is a case sensitive language. Most of you will say "So what, wasn't C?" Yes and no. C was case sensitive within the language, but converted all of its entry points to upper case. Some of you may remember my debugging tip about the debugger not being able to find your function names when you tried to set a break point typing in lower case.

Java is case sensitive outside of the compiler. When you declare a class it is customary to used mixed case names. When Java attempts to run your P-compiled class from within the VM (Virtual Machine), it looks for a file name matching the class name in exact case. You can either break tradition and work completely in upper case, or you must get used to putting quotes around your file names when using the tools. This pitfall exists on all stable and mature platforms where Java is ported. It isn't a problem on the lesser platforms.

Another major pitfall with Java is that it is an semi-interpreted language. Most call it an interpreted language, but I do not. You P-compile Java source code into a syntax the VM understands. The VM then interprets the P-compiled code at run time. While many will lament about the performance penalty of interpreting code, I won't bother. My issue with the VM is that you can't use it from within the safety and security of an ACMS application.

Yes, you can set up a C program or function to call Java on OpenVMS, but once you do that, you are running inside of the VM, not the controlled and secure environment of ACMS. At least that is my opinion. Should the Java VM die spastically without communicating its death back to the caller, your server will sit idle waiting for a response until it has reached the configured time out parameter. Then, and only then, will ACMS kill off the task and start a new server to handle the message/task queue entry which was being processed by the previously existing server. Admittedly, it is a small beef. We face the same problem when we are accessing a relational database other than RDB.

Page left blank intentionally.

Chapter 1

Basics of Java

1.1 A Little Bit of History and Concept

Don't worry, I'm not going to bore you to tears re-hashing the same history lesson every other Java book feels compelled to provide. Not my style. You can read about that for free on-line, or you can buy one of the oh-so-numerous books on Java which will burn an entire chapter covering the history.

First, let me ask for a show of hands, or whatever limbs you happen to possess, "How many of you consider yourself descent, good or seasoned C++ developers?" A very special form of Hell waits for you while learning Java. "How many of you consider yourself comfortable or good with pointers in either C or C++?" Ah, one or more chemical dependancies are in your future learning this language. Choose your poison now!

This may not come as a surprise to you, but I'm not the biggest fan of this language. Having spent way too many years coding in C and C++ on viciously underpowered platforms tweaking cycles out of loops, I got really good with those languages. (Okay, I'm a lot better with C than C++, at least according to the purists.) This language offended me in many ways. While I am somewhat functional with the language, I am by no means an expert, and avoid it when possible. (I'm still trying to figure out how I got talked into writing this book. It sure wasn't the publisher, because I'm self published!)

I think the biggest source of problems I and most other C/C++ programmers have with this language are rooted in the syntax and where the language came from. Java was not originally developed to be a Web language. That happened when marketing got involved. Java was originally developed to be an embedded systems language. The kind of system which exists in your microwave, VCR and later model automobiles.

This concept wasn't too bad at the time it was developed. Having done some near embedded development myself in the early days of pen computing under DOS (don't ask!), I can tell you that state of RTOS (Real Time Operating Systems) for embedded use was not good. Most shops had a collection of routines they kept using and tweaking with each new project, but commercial RTOS were way too expensive and not much cleaner. Almost everybody was hacking in assembler, with only the high end shops using C. (Ironically, that's pretty much the way it was when I last looked at the embedded market as well.)

In truth, the concept was phenomenal. Rather than bread boarding together a CPU, some RAM, an EPROM and some IO devices; there were going to be these prepackaged chips containing all of that along with a VM which understood Java. These single chip machines were going to be built in volume and used for everything because they would be cheap enough.

That really was a nice idea. It may even happen at some point. I just don't see it happening as long as there are enough grizzled old timers still working. Too many of them know how to hook together a 4-bit or 8-bit CPU, some RAM and some IO along with many years worth of hand tuned assembler. When they get done, it has a production cost