The Trainer's Friend
INCORPORATED

Creating Modern Business Computer Applications on z/OS

It's Not Rocket Science, But It's Not Easy

- ♦ **A look at many of the new facilities for applications and applications development on z/OS**

by Steve Comstock

The following terms that may appear in this paper are trademarks or registered trademarks:

Trademarks of the International Business Machines Corporation: WebSphere, z/OS, z/Architecture, zSeries, DB2

Trademarks of Microsoft Corp.: Microsoft, Windows, Windows NT, Visual Basic, Microsoft Access, MS-DOS, Windows XP

Registered Trademark of The Open Group: UNIX

Trademark of Sun Microsystems, Inc.: Java

Registered Trademark of Linus Torvalds: LINUX

Registered Trademark of Unicode, Inc.: Unicode

Trademarks held on behalf of World Wide Web Consortium: W3C, XHTML, XSL, WebFonts

Trademarks of the Apache Software Foundation: ASF, Apache.

Acknowledgement

I would like to especially thank Hunter Cobb of The Trainer's Friend, Inc. for his invaluable suggestions and input.

Any errors found here are strictly my own, and I appreciate any comments, suggestions, or corrections.

Introduction

☐ **While attending a session at SHARE, I was struck by how complex the parts covered in the talk were, to accomplish a particular task; the session talked about**

- ♦ **Hosting a web site on z/OS using the free HTTP server**

- ♦ **Creating web pages that use style sheets, JavaScript, DOM, and more**

- ♦ **Using a COBOL program running as a CGI to access DB2 data and returning data and an image in an HTML page**

- ♦ **The image is stored as a BLOB in a DB2 table**

This made me realize how cool the mainframe has become these days, but I also realized it takes some work to get there. I used to say "It's a piece of cake" in saying what it takes to get to the level of skill to do this. But the reality is, a lot of pieces have to come together to be able to bake a cake like this.

The recent turmoil in the economy has caused management to pause, even stop, updating the skills of their staff. Hopefully we're getting to the point where we can once again look to improve ourselves, our organizations, and our people.

In this paper, we examine these questions:

* What makes a good, modern business computer application? (what are the goals and properties from both a technical and a business perspective)

* What skills need updating or introducing to our employess so they are able to produce applications with these characteristics?

Goals of a Business Computer Application

❏ **A business decides to create a computer application for these reasons**

♦ **New function is needed to provide new or improved products or services**

✗ And this function is not available in a commercial off the shelf product

♦ **The ultimate goal is to improve the bottom line through improving at least one of these areas**

✗ Garnering additional customers

✗ Increasing sales to, and satisfaction of, existing customers

✗ Lowering costs of research, design, purchasing, production, test, marketing, distribution or other back office / internal functions

✗ Strengthening the connection (dare we say 'loyalty'?') between the work force and the organization, thus improving quality and productivity

❏ **The quality of any business computer application is measured by these criteria:**

♦ **Effectiveness and accuracy - it works correctly**

♦ **Ease of use - work gets done with minimal effort by the user**

♦ **Robustness - it does not fail**

♦ **Security - it does not allow inappropriate access to data**

Business Computer Application Styles

☐ **There are three styles, or models, used in business computer applications today:**

    ♦ **Batch - an unattended run processing large numbers of records, such as printing bills or paychecks, statistical analysis of inventory, and so on**

        ✗ Typically applied against one or more master files or data bases

    ♦ **Transaction based - high speed, simple, secure transactions such as finding out if an item is currently available from inventory or applying a payment against an invoice; usually involving an employee between the system and the user**

        ✗ Each transaction usually carries a small amount of information, as characters

    ♦ **Web based - possibly lower speed, often interacting directly with the end user, usually browser based, such as on-line shopping carts, or exploring products and services descriptions**

        ✗ Page displays may include images, audio, media, and more

---

Skills Needed To Develop These Applications

❏ **These kinds of applications require the following skills in the developers**

♦ **Design skills to ensure the application will be easy to use, understand, and maintain**

  ✗ With specialites in user interfaces for interactive applications, as well as security, networking, perhaps end user training

♦ **Programming language skills, from the language(s) of choice**

  ✗ Classic: Assembler, COBOL, PL/I, C, C++, CLIST, REXX, SQL

  ✗ Newer: Shell script, HTML, XML, XHTML, Java, Perl, JavaScript, php, Python, Ruby, and many more

  ✗ Leading to a need to understand how to communicate across multiple languages when required

♦ **Programming functional skills: how to accomplish the various application sub-functions in the languages(s) of choice**

  ✗ Dependent on the environment and style of the application, but may include APIs for z/OS services, Language Environment (LE) services, CICS commands, z/OS UNIX kernel services, DLL linkages, Dialog Manager services, handling code pages, and more

♦ **Skills in coding and testing, such as ISPF and / or workstation based products such as Eclipse, JCL, versioning software**

Skills Needed To Develop These Applications, 2

❑ **At a high level, then, here are some of the skills the developer needs to acquire in order to design / implement / maintain the kinds of business applications we are talking about on the mainframe:**

♦ **Code pages and Unicode: interface with data from non-EBCDIC systems**

♦ **ISPF skills: new productivity facilities, new Dialog Manager developer facilities**

♦ **z/OS UNIX: basics, file systems, commands, available callable services**

♦ **Markup languages: for example: HTML, XML, XHTML**

♦ **Web basics: X/HTML, JavaScript, CSS, DOM, CGIs, cookies**

♦ **Language Environment: concepts, available callable services**

♦ **Programming Languages: current developments in Assembler, COBOL, PL/I, C/C++; inter-language communication**

♦ **Programming Languages: newer choices such as Java, perl, php, python, etc.**

♦ **DB2: latest facilities such as Unicode host variables, XML data type and built-in functions, native SQL stored procedures, enhancements to SQL, improvements for working with LOBs**

♦ **CICS: CICS/TS and the ability to create web-enabled CICS programs**

Thare are others, too, but the point is: in order to use your z/OS system to create world class business applications, you need to make sure developers have a chance to learn the relevant skills in a fast, effective way. If you are in management, that is one of your charters.

The remaining pages here provide the next level of detail on the above topics; if you are not technical, feel free to skip right to page 20 for the summary.

Code Pages

☐ **We begin with the need to understand how code pages work and influence design and implementation, since this understanding pervades all new development**

A code page is a formal enumeration of what bit patterns map to what characters (and vice versa: it must be a two way street).

The IBM mainframe has classically used a code page called EBCDIC (Extended Binary Coded Decimal Interchange Code) which maps the basic Latin character set in a certain way. But this is misleading, since it turns out there are numerous EBCDIC code pages! This came about due to the need to add characters not found in the English alphabet. So there are EBCDIC code pages for USA and Canada, Finland and Sweden, Denmark and Norway, and so on.

On the other hand, the rest of the industry pretty much relies on ASCII code pages (American Standard Code for Information Interchange). But wait: there is more than one ASCII code page.

Still other code pages arose around the world to meet local needs. It has become a real Tower of Babel issue: how does a system display any particular bit pattern so it can be viewed correctly by an end user? The answer, unfortunaltely, is "It depends on what code page was used to create the string of bit patterns".

Unicode

One positive development here is the creation of a standard called Unicode: determined to be a Universal Code Page. It provides enough bit patterns to hold a unique representation for every character in use by every human language. If every character string is encoded using Unicode, then there will never be a problem determining how to display or interpret a bit string, nor how to encode a character.

Unicode started out as a 16-bit encoding: each Unicode character takes two bytes. This allows for over 64,000 characters. Plenty, you might imagine. But not really! The most recent version of the standard, 6.1 (February, 2011) includes 109,384 assigned characters and code points.

Handling this large number of characters is accomplished by having some characters assigned to two 16-bit patterns. There are three basic formats of Unicode encoding: UTF-8, UTF-16, and UTF-32, each with their own pros and cons.

Unicode is gaining acceptance, and it is now part of the standards of how we should encode XML, XHTML, and HTML, for example. Java character strings are Unicode strings.

So today's application developer needs to be aware of issues surrounding code pages, and how to handle data encoded in various code pages in any given application

Today's IBM mainframes include hardware instructions to work with data in ASCII and Unicode, as well as the classic EBCDIC. So the Assembler programmer has to be aware of these. COBOL, PL/I, and C all support Unicode data with language-specific functions and compiler options. A set of system services for Unicode work provides a suite of APIs for handling Unicode.

❏ These are issues the traditional application programmer seldom worried about, but which need to be faced in today's world

---

ISPF

☐ **ISPF (Interactive Structured Programming Facility) is a full screen front end to TSO (the Time Sharing Option)**

 ♦ **As well as its own application development, deployment, and runtime environment for locally hosted TSO-based applications**

☐ **ISPF has grown in capablity and ease of use over the years and is still the premier window on z/OS for most application developers**

 ♦ **In order to be more productive, developers trained on ISPF more than five years ago almost certainly need quick update information on the new facilities available such as**

 ✗ Multiple logical screens (up to 32) along with screen naming and a point-and-shoot list of screens for rapid swapping

 ✗ Keylists

 ✗ New options for allocating data sets

 ✗ Expanded facilities of DSLIST (option 3.4)

 ✗ Ways to copy / move data without pre-allocating the target

 ✗ Scrollable panels and scrollable fields

 ✗ Alternative ways to recall previous commands

 ✗ Enhanced editor features including

 ➢ Line commands: (,), >, <, AK, BK, LC, OK, HX, UC

 ➢ Primary commands: CUT, EDSET, FLIP, HIDE, PASTE, UNDO

 ➢ Other: browse, edit, view ASCII and Unicode data

 ✗ How to use referral lists

 ✗ UDLIST (option 3.17) to work with z/OS UNIX files

 ✗ In some cases, Dialog Manager enhancements

---

z/OS UNIX: Running UNIX on your z/OS System

☐ **In the 1990's, IBM announced something called "MVS Open Edition" or MVS-OE**

- ◆ **This was a set of facilities that enabled UNIX facilities on MVS systems**

- ◆ **In fact, at one point MVS was "UNIX-branded" meaning it had passed all the tests / criteria to be officially considered a certified UNIX platform**

☐ **MVS-OE has changed names several times; the current name is "z/OS UNIX System Services"; the official short name is "z/OS UNIX"**

☐ **Although over the years it has grown in rigor and capacity, many installations have had bad experiences with this component, resulting in a reluctance to use it for anything other than the minimal necessary tasks (such as FTP and TCP/IP)**

☐ **But z/OS UNIX is one of the best ways to exploit the modern mainframe at low cost:**

- ◆ **z/OS automatically comes with a free HTTP server!**
  - ✗ Runs under the z/OS UNIX shell
  - ✗ Serves webpages stored in z/OS UNIX files
    - ➢ Including images, PDFs, audio, video, and other media

- ◆ **In addition, a free Apache-based HTTP server is also available**
  - ✗ It's no charge, but it must be explicitly ordered
  - ✗ Same kinds of capabilities as the other free server, but a little more modern and closer to the open source model

---

z/OS UNIX: Running UNIX on your z/OS System, continued

☐ **z/OS UNIX facilities include a UNIX file structure consisting of directories and subdirectories and files, locating files through paths and various environment variables**

☐ **Java runs on the mainframe, under a z/OS UNIX shell, providing all the power of that language**

☐ **This all allows classic z/OS (MVS) programs running as before, on the same system running UNIX applications using z/OS UNIX facilities at the same time. Cool!**

　♦ **Cooler: classic programs can access files in the z/OS UNIX file structure, and programs running under z/OS UNIX can access traditional z/OS (MVS) files**

　　✗ With a few restrictions in both cases

☐ **Furthermore, most z/OS UNIX kernel services are callable APIs, so you can avail yourself of these services from Assembler, COBOL, PL/I, C, and more; services such as**

　♦ **Dynamically loading programs from the z/OS UNIX file system or a z/OS library**

　♦ **Processing z/OS UNIX data files**

---

z/OS UNIX: Running UNIX on your z/OS System, continued

❑ **So the application developer on z/OS who will be using these facilities needs to learn**

- ◆ **How to get to a z/OS UNIX shell (omvs under ISPF or telnet from outside the system)**

- ◆ **How the file and directory structure is organized**

- ◆ **How to copy, move, and change attributes of files in the z/OS UNIX file system**

- ◆ **How to access, update, create environment variables**

- ◆ **At least a basic set of shell commands - more when developing applications that will run under the shell**

- ◆ **How to edit files in the z/OS UNIX file system (oedit or vi)**

- ◆ **How to copy files from the z/OS UINX file system to z/OS MVS files**

- ◆ **How to copy z/OS MVS files into the z/OS UNIX file system**

- ◆ **How to run programs under the shell**

- ◆ **How to submit jobs to the batch from the shell**

- ◆ **How to create and run shell scripts**

Markup Languages

❏ **The concepts of marking up text to indicate how the text should display have a long tradition**

♦ **Which has grown and expanded to include these markup languages that today's application developer should have some familiarity with:**

✗ HTML - HyperText Markup Language: presentation of text and other media, depending on the "user agent" (browser or audible reader or Braile printer and so on)

➢ As well as the concepts of linking to additional files and pages

✗ XML - eXtensible Markup Language: allows user-defined tags and attributes under strict composition rules; fast becoming the most common way to store and transmit data

✗ XHTML - eXtensible HTML: an XML definition of HTML

Note that these should usually be encoded in Unicode!

Markup Languages, continued

☐ **There are also supporting technologies for these markup languages that should be learned, among them:**

- ♦ **CSS - Cascading Style Sheets: a facility for implementing external rules for formatting, so markup can be mostly concerned with content while presentation is directed by these external style sheets**

- ♦ **DOM - the Document Object Model: a rigorous way to access parts of a document based on the markup**

- ♦ **XSLT - XML Styleheet Language for Transformations: not only can specify presentation but also alternative ways of viewing and organizing the same data**

Web Basics

☐ **People who will be developing HTML pages to be served from your z/OS system need to know these techniques and skills:**

♦ **How to code HTML and XHTML including facilities such as**

    ✗ Anchors and links

    ✗ Embedded style markup, embedded style sheets, and external style sheets

    ✗ Embedded images, client-side maps, and multimedia objects

    ✗ Client side scripting such as JavaScript

    ✗ Using DOM (the Document Object Model) to access, change, insert, and delete document nodes under script control

    ✗ Forms and controls to gather and display information

    ✗ Cookies and hidden controls

    ✗ Tables

    ✗ Framesets

♦ **How to implement CGI (Computer Gateway Interface) programs to process requests from forms and respond with dynamically constructed HTML pages**

    ✗ Coded in various languages

☐ **Note that programmers are not generally good designers of web pages from an aesthetic / artistic point of view: you will need people with special skills for that kind of work**

Language Environment (LE)

❑ **This component of z/OS provides a single run time environment for all high level languages (HLLs), especially COBOL, PL/I, and C/C++**

♦ **These compiler produce "LE-conforming" object code**

♦ **Assembler programs can be made LE-conforming**

❑ **LE also provides a suite of callable services available to all LE-conforming programs**

♦ **Knowledge of many of these services can be useful for application developers**

❑ **LE enables / supports z/OS UNIX**

❑ **Since all HLLs use the LE runtime, it is possible, for example, to call C functions directly from programs coded in COBOL, PL/I, or LE-conforming Assembler**

♦ **This greatly expands the range of easily useable, pre-coded functions**

✗ For example, the printf and scanf functions may be easily invoked from any LE-conforming program

❑ **Furthermore, calling among programs written in different languages is greatly simplified, since they do share this common run time**

Programming Languages

☐ **All the common, classic programming languages have undergone extensive changes over the last dozen years**

Programmers not familiar with the new features continue to code using their old styles: that's human nature. But in doing so they miss the opportunity to create applications using the new paradigms. For example, in the modern z/OS world, programs in all languages:

- ♦ **May be written in mixed case**

- ♦ **Can process ASCII and Unicode**

- ♦ **Can parse and generate XML files**

- ♦ **Can be DLLs (Dynamic Link Libraries) and may invoke DLLs**

- ♦ **Can create and use LE condition handling instead of having to code Assembler ESTAE / ESPIE routines**

- ♦ **Can access files in the z/OS UNIX file space**

- ♦ **Can run under the z/OS UNIX shell**

- ♦ **Can use callable services of: z/OS UNIX kernel, LE, Unicode services, XML services, TSO and REXX services, and more**

- ♦ **Can dynamically allocate files at run time**

Programming Languages, continued

❑ **Languages more traditionally found in the UNIX world have mostly been ported over to z/OS (running under the z/OS UNIX shell)**

♦ **These languages include: Shell script, Java, Perl, JavaScript, php, Python, and Ruby**

❑ **The ability to use these languages, in and of itself, might be useful in any number of situations**

♦ **And since each of these have libraries of useful functions already available, development time may be shortened and application quality may be improved**

DB2

☐ **The last two releases of DB2 have been especially significant in scope, and you want your developers to know how to exploit all the new functionality**

♦ **Some of the more significant changes include:**

✗ SQL data manipulation language changes: scrollable cursors, rowsets, EXCEPT, INTERSECT, MERGE

✗ Character data in the DB2 catalog is encoded in Unicode

✗ New data types: XML, BIGINT, DECFLOAT

✗ The use of the Resource Recovery Services Attachment Facility (RRSAF) for communicating with DB2; this is particularly well-suited to the CGI environment

✗ DB2's ability to act as both a web services consumer and a web services provider

✗ Native SQL stored procedures

✗ Call level interface (ODBC), plus JDBC / SQLJ for handling data base requests from Java programs

✗ XML Schema support

                   Business Applications

CICS -> CICS/TS

❑ **The evolution of CICS to CICS / Transaction Server introduces new programming techniques and capabilities, including**

♦ **Web-aware CICS programs**

♦ **CICS as a web server**

♦ **Document templates for web-aware applications**

♦ **CICS as a web services requester**

♦ **CICS as a web services provider**

Summary

❑ **So if you want to be able to field applications in this new world, you need to get your z/OS developers up to speed; this requires:**

♦ <u>**A plan for each developer**</u>**, some combination of self-study, mentoring, classroom training, seminars or conferences, and other delivery methods concentrating on the areas discussed here**

✗ Tailored to each individual (or at least a group or team), with concrete skill targets and deadlines

♦ <u>**Commitment to the plans**</u>**: allowing people time for learning, budgets that pay for training (or reimbursements for out of pocket costs incurred)**

❑ **A note about standards based topics:**

♦ **Unlike IBM-specific areas such as DB2 and the IBM Enterprise compilers, people who need to learn topics that are based on standards from international standards organizations will need to learn where to find resources**

✗ W3.ORG, IETF.ORG, Unicode.org, and other sites provide the source material, instead of IBM manuals

✗ Most publications and tutorials in these topic areas tend to not be aware of z/OS, IBM mainframes, and EBCDIC kinds of issues, so you may need to search wider than others do

✗ These documents also tend to be very turgid and hard to read, so buying some books from the technical book stores is often heplful

Summary, 2

- ❑ **The Trainer's Friend is a training organization focusing on applications development for the z/OS environment**

    - ♦ **With a preference for classroom based delivery**

- ❑ **We offer training and other resources that are designed to help developers quickly become adept in the new technologies under z/OS**

    - ♦ **You will find our list of these offerings at www.trainersfriend.com/modern**

- ❑ **There are other providers available too, of course, and we urge you to compare and select the offerings that best meet your needs, whatever they may be**

- ❑ **Good luck in the brave new world.**

This page intentionally left almost blank.