Upgrades: PowerBuilder 8 – Poised to Conquer  
And on the eighth release...

PBDJ Feature: PowerBuilder 8 for the Rest of Us  
New functionality for “traditional” client/server developers

PB Wizards: PowerBuilder 8 Web Targets  
Improve development productivity

From PowerBuilder to Java: The (New) Exception Call Stack  PART 1 OF 2  
Guess what? PowerBuilder is starting to look like Java

PBDJ Feature: PowerBuilder 8.0 and EAServer Integration Improvements  
Think of it as cloning without all the overhead

PowerDesigner: Charting the Flight of a Killer App  
A migrating exercise shows some PB 8 virtues

PBDJ Feature: Exception Handling in PowerBuilder 8  
Creating more robust applications
e crane

www.ecrane.com/testdrive.com
Dedicated to PowerBuilder 8
Once I make the switch … I’ll never go back.

It’s been a long time since we’ve had a new version of PowerBuilder. I don’t know about you, but I’ve been very patient and can’t wait any longer. I’m particularly excited about PowerBuilder 8 because many of its new features address the type of development I’m doing – building Web applications consisting of JSPs calling PowerBuilder components running on EAServer.

PB7 had several wizards, which helped build and deploy components as well as build proxies. However, when the PB7 team was designing it, they didn’t have the benefit of input from hundreds of active developers. PB7 development for EAServer was new territory and no one could foresee the exact ways PB would be used.

Right out of the gate some deficiencies became apparent and many of those were fixed in patches to PB and subsequent versions of EAServer. Now, nearly two years later, a lot of PB development for EAServer has been done and some development techniques have become de facto standards. For example, it’s common for developers of PB components to use one application per component. This causes some hassles and frequent application switching during development, but results in smaller and faster components.

Unlike PB7, the PB8 designers had the benefit of input from experienced PB component and Web developers. It shows. One main focus was to make PB a better tool for developing components – that means better features, better integration to EAServer, and a better development environment. The hassle of frequent application switching is going away because the IDE has a new paradigm: workspaces and targets. In PB8, components have their own context and you don’t have to worry about switching applications in order to edit an object, any object. For more on this subject, see the article on workspaces and targets by Larry Cermak.

The IDE change is just one example of what PB8 brings us. Though we can’t cover all the features in one issue, we’re going to do our best. This issue is dedicated to PB8 and is loaded with articles by the experts you’ve become accustomed to reading. We have entire articles delving into some of the most important new features – plus two overview articles. One covers the entire list of new features (see Tom Peters’ article) while the other focuses on those features beneficial to client/server developers (see Bruce Armstrong’s article). If you are still developing client/server applications, don’t worry. You aren’t being left behind. PB is still a great client/server development tool ... though it is also much more.

Making the switch to PB8 can be likened to when I bought my first microwave oven. Before I got the oven, I was in the habit of heating all my food in the oven or on the stove. After I got into the habit of using the microwave, I didn’t want to go back. Why should I spend 10 minutes heating water on the stove when I can do it in one minute in the microwave? Once I make the switch to PB8, I’ll never go back.

In the coming months, we’ll continue to have great articles about new PB8 features, but we will also focus on other topics. Next month, we’ll focus on training – and the following month on TechWave. After that it’s back to new releases as our issues will coincide with major product releases that are fast approaching. ▼

Author Bio
John Olson is principal of Developer Inc., a consulting company specializing in software solutions using Sybase development tools. A CPD professional, Sybase-certified instructor, and charter member of TeamSybase, he is a coauthor of SYS-CON’s Secrets of the PowerBuilder Masters books.
Two years ago, I interviewed Sybase’s Bill Bartow – then vice-president of marketing and engineering for the Internet Applications Division. Among the things we discussed were Sybase’s plans for PowerBuilder 8 (PB 8). Now that PB 8 is here, I’d like to look back at that interview and see how closely the product tracked with those plans.

At the time, Bill made the following claims:

- **With PB 8 and beyond the goal would be higher quality.**
  Personally, I’d say that goal has been met. With a product as multifaceted as PowerBuilder, I’m sure a few bugs have slipped through. However, I can say that I was much more pleased with the quality of the builds during the PB 8 beta process than I have been with several of the previous product version beta cycles. I’m sure this will be reflected in the quality of the officially released product.

- **PB 8 would provide a common approach for viewing and defining projects and workspaces.**
  Another goal met, well beyond my expectations at the time. Not only has PowerSite been integrated into the PowerBuilder IDE, but so has a good portion of the functionality of Jaguar Manager. One of the major gripes I have with PB 7 is switching back and forth between several different IDEs. With PB 8 that has largely been eliminated.

  Some more work could be done with Jaguar Manager integration. I still need to use it – and it still must be installed on NT – to do things like generating stubs or TLB/REG files. However, I believe that the new “workspace” concept is one of the most significant ease-of-use enhancements in the product.

- **PB 8 would include significant new features.**
  Another goal met. Perhaps nothing as significant as the introduction of object-orientation with PB 3, PFC with PB 5, or the HTMLDW with PB 7, but quite a number of important new features nonetheless. Obviously, many of those features were aimed at the n-tier and Web development folks. However, for those of us who are still spending the majority of our time working on “traditional” client/server applications, PB 8 offers quite a few items of interest as well. I discuss this in more detail in a separate article in this issue of *PBDJ*.

One thing Sybase did differently during the PB 8 development cycle was – in conjunction with the International Sybase User Group (www.isug.com) – to open up the list of enhancements that are being considered so the user community could vote on them. I think that process was largely responsible for the significant number of enhancements in the final product that will make my day-to-day work much easier. To that end, I encourage Sybase to do the same during the PB 9 development cycle.

The one downside, though, is that the voting occurred about a year and a half ago. I don’t think people had any idea then of how important technologies such as XML would become in the interim. As a result, PB 8 doesn’t have a XMDLW. You can be sure that it’s near the top of my list for PB 9 enhancements. Perhaps Sybase should allow the PB 9 feature list to be a bit more fluid during the development cycle. That is, if significant new technologies evolve during the development cycle, Sybase should try to incorporate them into the final product.

In any event, I’m looking forward to working with the new product, and I hope you are as well. See you at TechWave 2001!

**Author Bio**

Bruce Armstrong is a senior PowerBuilder developer with the Los Angeles office of Kforce.com. A member of TeamSybase, he has been using PowerBuilder since version 1.0. He was a contributing author of *PowerBuilder 4.0 Secrets of the Masters*. 

**WRITTEN BY**

Bruce Armstrong
Get Up to Speed with the Fourth Wave in Software Development

- Real-World Web Services: Is It Really XML's Killer App?
- Demystifying ebXML: A Plain-English Introduction
- Authentication, Authorization and Auditing: Securing Web Services
- Wireless: Enable Your WAP Projects for Web Services
- The Web Services Marketplace: An Overview of Tools, Engines and Servers
- Building Wireless Applications with Web Services
- How to Develop and Market Your Web Services
- Integrating XML in a Web Services Environment
- Swing-Compliant Web Services
- Real-World UDDI
- WSDL: Definitions and Network Endpoints
- Implementing SOAP-Compliant Apps
- Deploying EJBs in a Web Services Environment

The First Major Conference & Expo Focusing Exclusively on Web Services

Integrating applications with Web Services
Web services, in its infancy, promises to be the next major impact on the business model of the new millennium.

The Web Services Edge Conferences in New York and California provide the venue to explore the implications of Web Services and their impact on business process.

The Expo will bring together leading technology vendors and their business partners to showcase the products and services necessary to integrate applications and create e-business applications.

Now is the time for companies to become educated on the opportunities Web Services create for more efficient enterprises.

Web Services Edge is the venue to get up to speed.

Plan to Sponsor
Many sponsorships are exclusive. Rise above the noise and let SYS-CON Media's integrated marketing work for you.

Plan to Exhibit
Call 201 802-3004 to reserve space

Steve Benfield, a leading industry expert and frequent speaker at business and technology events, is also cofounder and CEO of Bondi Software.

Steve Benfield
Conference Tech Chair

OWNED AND PRODUCED BY

WEB SERVICES EDGE

© 2001 SYSCON EVENTS

135 Chestnut Ridge Road Montvale, NJ 07645
201 802-3069
Fax: 201 782-9601
visit
WWW.SYSCON.COM/WEB SERVICESEDGE
for more information

NEITHER SYSCON MEDIA NOR SYSCON EVENTS, INC., ARE SPONSORS OF OR ARE AFFILIATED IN ANY WAY WITH, CAMELOT COMMUNICATIONS, INC., OR ANY EVENTS PRODUCED BY CAMELOT COMMUNICATIONS, INC.

MEDIA SPONSORS:
No time for anecdotes or attention-grabbing illustrations! This new version of PowerBuilder is so packed full of great new features that I haven’t got time for trying to catch your interest. Let’s just say that PowerBuilder 8 is the best version to date.

The IDE has evolved once again and will prove to be most efficient for all developers, whether you’re strictly client/server or utilizing EAServer to the full. New objects are allowing us to build far more robust applications than we ever imagined. New File and Printer system
functions are reducing the need for API calls. There are new DataWindow expression functions and miscellaneous new functions. It goes on and on.

The purpose of this article is to give you an informative, yet concise overview of PowerBuilder 8.0, new features. In doing so I hope to prove to you that this new release is the best to date and how much you, as the developer, stand to gain by upgrading to 8.0. In order to keep this article to a reasonable length and, more importantly, to avoid stepping on the toes of the other authors covering specific features, I will be focusing on IDE and PowerScript improvements, with brief references to other areas.

New Look, Same Feel, Different Places

If you've been using PowerBuilder 7, then you're used to working in panes. PowerBuilder 8 follows the same feel, but it looks substantially different thanks to some new features. Take a gander at Figure 1 and you'll see what I'm talking about. All of these panes are resizable and moveable. (Sybase refers to these as windows. I use the terms 'pane' and 'window' interchangeably to keep consistent with the IDE schema, or my stubbornness… you decide.) Just as in PB 7, you can click and drag just about anything. You will notice some new panes, toolbar buttons, as well as some missing toolbar buttons. There has been no removal of any 7.0 functionality, just some access relocations. At first, I was afraid that this would entail more of a habit-changing effort than it turned out to be. The new location of old functionality actually lends to the greater intuitiveness of the interface.

I'll give you a for-instance before getting into each part of the IDE. Because the IDE is now workspace- and target-driven, there are a few items that have been removed from the menus and toolbar. As I stated, the functionality is still there, but you need to know how to use the System Tree pane (Figure 2). You'll notice that you begin in a Workspace (first tree item PBDJ) and then a Target (second tree item PBDJ). In this example, our target is an application that I created using the Template Application Wizard. Under this target you can see all of your PBL and object entries. With this new System Tree pane, you no longer need the Library List button. Why? Because a PBL is either in your Target or not and you can see which libraries are in your Target via the System Tree pane. You right-click the Target to get to its properties. That's where the Library List rightfully belongs. You also open Workspaces and Targets, not applications. No more Select Application menu item!

Changes like these run throughout the IDE. Trust me, the learning curve is slim to none. Now let's talk about the System Tree.

The System Tree Window

This pane can be hidden and shown at will by clicking the System Tree toolbar item. You'll notice in Figure 2 that the System Tree window has four tabs. These are, in order, the Workspace, Page, Language and Components. To stay within the scope of this article, we only need to see the Workspace tab.

Starting with the root of the tree, the layers shown are 1) Workspace, 2) Target, 3) PBL, 4) Object, 5) Object Properties, Methods, etc. You may only work within one Workspace at any given time. You can have multiple Targets in one Workspace, though, and work on these concurrently. There are two Target types, PowerScript and Web. Figure 2 shows a PowerScript target, a run-of-the-mill application.

There's a lot of great functionality built into this pane. Some of which will save you from having to jump into the Library Painter or the Browser. As you can see, Properties, Events, Functions, Structures and Controls can all be viewed from here. By right-clicking on an object, you can perform expected functionality such as Edit. You can also perform a lot of functions that previously had to be done through the Library Painter. From the System Tree pane, you can Copy, Move, Delete, Regenerate, Export, Search, and Print your objects.

A piece of functionality that's brand new in 8.0 is the ability to edit exported object source right in PowerBuilder. That's right, no more going to the Library Painter, exporting the object syntax, opening the File Editor, modifying your syntax and importing the object back into your PBL. Now you simply right-click your object and select Edit Source. A script painter opens where you can make your changes. Clicking the Save toolbar button recompiles your syntax into the PBL. Don't worry, you can still Export and Import syntax as you've been used to. (Per Sybase, use this feature with caution. If you run the risk of destroying your
object editing source outside of PowerBuilder, it's even easier here. If you make a mistake and save it, the validation will still stop you from closing the painter until you've corrected the errors.) By right-clicking on a PBL, you have all of your standard Library Painter functionality such as Optimize, Build Runtime Library, etc.

Another great new feature that resides within the right-click popup menu is Inherit. You don't need to click the Inherit button on the toolbar anymore. Now you just select the object you wish to inherit from and choose Inherit from the popup. Instantly, and I mean instantly, your new descendant is opened and waiting for you to code. For window and DataWindow objects, the popup menu also has a Run/Preview function, eliminating the need to use the Run/Preview toolbar item.

In explaining the System Tree, I've left what I view to be the most powerful features for last. By right-clicking a Target or a Workspace, you get the New menu option which opens the New wizard allowing you to create and add your new Targets, objects, and so forth. Also available here are the options to do Full or Incremental Rebuilds at the level you've selected. So if you have several PowerScript Targets in your Workspace, you can do as full a Full Rebuild as you know how! This is quite fast, too. I realized a 20% improvement in the time it takes to do a full rebuild on one of my larger applications between 7 and 8. Something nice that's been added to the rebuild functionality is the ability to Stop the rebuild operation completely without crashing out of PowerBuilder. You still may need to regenerate, but you're not forced to perform a migration. Sweet!

That's all I can cover for the System Tree pane. One point worth mentioning is that, as usual, any functionality found in popup menus can be found in the IDE menu. If I had to come up with one area for improvement in this pane, it would have to do with the links to any object painters. In PB 8, you can jump right into an object by selecting Edit, which is nice, but this is the same as we had in the Library Painter in 7.0. What I'd like to see is the ability to select an object method or property from the System Tree and jump right into that specific method or property for editing.

The Clip Window

The Clip Window is a brand new feature that will make coding a lot easier for us. It's basically a repository for frequently used scripts. For instance, let's say you have a standard piece of error-handling code. Select the code in a script painter, then Copy and Paste it into the Clip Window (see Figure 3). You can also drag and drop an object from the System Tree into the Clip Window. This doesn't copy the actual object. It copies the class name of that object. In the Clip Window, you name your code something meaningful so you know what you're grabbing later. If you look at Figure three, you'll see I've got three pieces of code that I've pasted in there, naming them SQLCA_Connect, SQLCA_Disconnect, and SQLCA_Error. I can rename these as needed by clicking the Rename button in the Clip Window.

Using the scripts is just as easy as loading them into the Clip Window. Simply select the script you want to use and drag it into the script painter in which you're working. When you're done with that piece of code you can remove the entry from the Clip Window.

Slick, slick, slick! Although I fear this may hurt some third parties that offered tools to do this in previous versions of PowerBuilder. One improvement I'd like to see in the Clip Window is the ability to sort your entries. Right now, they remain listed in the order in which you loaded them. You cannot load a new entry before or between existing entries. Another point worth mentioning is that these entries are not Workspace-specific. Regardless of where you're working your entries stay with you. That's either a benefit or a detriment depending on your development style.

The Output Window

If you look at the bottom of the IDE shown in Figure 1, you'll see the output window running the length of the screen. Just like the System Tree and Clip windows, the Output Window can be hidden and shown by clicking its corresponding toolbar button. This pane shows any and all output from developer driven operations. If you do a rebuild, you'll see the results in the Output Window. The same holds true for any Search operation you perform for, let's say, your entire workspace. So on and so forth.

Once you've generated your output, you may want to review it. Rather than scrolling through the screen searching for the exact result or error you're interested in viewing, you can use the Next Error/Message and Previous Error/Message buttons on the toolbar. Of course, you can access the same functionality by right-clicking the Output Window. If you so choose, you may also Save the results from the Output Window, jump right from your Selected Error/Message into the script painter by selecting Edit, or select Edit Source and edit the object syntax. You can also Print the results from the Output Window and/or Clear the window when you're done.

The IDE Now Has a Memory of Its Own

This is going to make any PowerBuilder developer happy. Starting in version 8.0, you now have the options for PowerBuilder to open right to the place you want without coding start-up switches. You have three options you may select in this area, which reside on the Workspaces tab in the System Options dialog.

First, you can opt to have the startup dialog greet you with no workspace being loaded. This affords you several opportunities at start-up. When firing up PowerBuilder, the Startup Dialog offers you the options 1) Create a New Workspace and a New Target, 2) Create a New Workspace and Add an Existing Target, 3) Create just a New Workspace, and 4) Open an Existing Workspace.

Second, you may opt to Reopen Workspace on Startup. It's just what the name implies. Whichever Workspace you were in when you last closed PowerBuilder will be the one automatically opened upon start-up.

Third, you may opt to Reload Painters when Opening Workspace. You have to heed the nomenclature used for this option. It doesn't read "Reload Painters at Startup," it reads "Reload Painters when Opening Workspace." This means that as you switch between Workspaces while keeping PowerBuilder open, the painters you had opened the last time you were in that Workspace are reloaded for you.

These are great new features, particularly the automatic reloading of your painters in the Workspace. However, one thing I discovered while testing this was that opting to have the Startup dialog show at startup is mutually exclusive to the Reopen Workspace on Startup option, yet you can select both simultaneously. The verbiage of each option logically excludes the other, but it could be confusing as to what you may expect to see when starting PB.

One last feature on this tab is the Clean Up button. PowerBuilder uses the system registry to track Workspace and Target information. By clicking the Clean Up button, PowerBuilder removes any orphan entries from the registry.
SYS-CON Custom Media is a new division of SYS-CON, the world’s leading publisher of Internet technology Web sites, print magazines, and journals.

SYS-CON was named America’s fastest-growing, privately held publishing company by Inc. 500 in 1999.

SYS-CON Custom Media can produce inserts, supplements, or full-scale turnkey print magazines for your company. Nothing beats your own print magazine for sheer impact on your customers’ desks...and a print publication can also drive new prospects and business to your Web site. Talk to us!

We work closely with your marketing department to produce targeted, top-notch editorial and design. We can handle your distribution and database requirements, take care of all production demands, and work with your marketing partners to develop advertising revenue that can subsidize your magazine.

Do you need to differentiate yourself from your competitors?
Do you need to get closer to your customers and top prospects?
Could your customer database stand a bit of improvement?
Could your company brand and product brands benefit from a higher profile?
Would you like to work more closely with your third-party marketing partners?
Or would you simply like to be a magazine publisher?

So contact us today!
East of the Rockies
Robyn Forma
robyn@sys-con.com
Tel: 201 802-3022

West of the Rockies
Roger Strukhoff
roger@sys-con.com
Tel: 925 244-9109

Your Own Magazine

2 years worth of...
Features & Articles
Product Reviews
Case Studies
Tips & Tricks
Interviews
Editorials
IMHOs
& more!

Check out over 150 articles covering topics such as...
Questions? E-mail XMLCD@SYS-CON.COM

The Complete Works

only $49.99

Searchable HTML Format

2 years worth of...
Features & Articles
Product Reviews
Case Studies
Tips & Tricks
Interviews
Editorials
IMHOs
& more!

Check out over 150 articles covering topics such as...

Questions? E-mail XMLCD@SYS-CON.COM

Printable HTML format

2 years worth of...
Features & Articles
Product Reviews
Case Studies
Tips & Tricks
Interviews
Editorials
IMHOs
& more!

Check out over 150 articles covering topics such as...

Questions? E-mail XMLCD@SYS-CON.COM

PDF files

ORDER ONLINE AND GET 10% DISCOUNT GO TO WWW.JDJSTORE.COM TO ORDER
AutoScript Enhancements

Sybase has made a lot of nice improvements in the coding department of PowerBuilder. Aside from some new functions, which I’ll touch on later, they’ve improved the interface features to make coding even easier. This is an area that’s starting to show up my years in the industry. I’m beginning to think things like “developers today don’t know what it’s like to really code. When I was doing COBOL I had to type everything. No painting your screens, no drag and drop coding. Now, even the 4GL IDE’s are making coding logic a luxury…” Enough said.

As though we needed to get any lazier, Sybase has afforded us the opportunity to do so if we choose. AutoScript was introduced in version 7 in order to make our coding efforts less taxing. Not only does it save time coding, it also ensures against incorrectly typing method names and/or using incorrect arguments. We no longer needed to lookup methods in other painters or the Browser. Any and all methods and variables that belonged to the object preceding the dot got listed for us. Sybase recognized that there were some areas for improvement and have made the changes.

In version 7, AutoScript was either on or off. Regardless of where or when you were coding, if AutoScript was on, the popup box would bug the life out of you as you tried to knock out code you had memorized. Starting with 8.0, you can now flick AutoScript on and off with a shortcut key. Shortcut keys are not new in 8.0. They were introduced in 7, but they were limited when it came to menu items like AutoScript. Because AutoScript was buried in the Design… Options… dialog, you couldn’t activate or deactivate it with a shortcut key. Thanks to improvements in the Shortcut Keys functionality, you now can. Activating and Deactivating AutoScript has also been added to the popup menu in the script painter.

New to PB 8 is the ability to specify what appears in the AutoScript popup. The AutoScript dialog (Figure 4) has changed to allow you to select exactly what should appear and how it should be resolved. If you select to only see methods, then you’ll never get properties, variables, etc. What’s even better, is that with Partial Name Resolution, the list in the AutoScript popup shrinks to include only entries matching what you’ve typed. If you recall, PB 7 showed everything and then scrolled to the closest matching entry and didn’t do too good a job of it. Sometimes the AutoScript popup would get cropped and you couldn’t even see which entry was highlighted. Also, if there is only one matching entry that could complete what you’ve typed, it’s automatically painted into the script for you. For instance, if there is only one method that starts with “of_Get” in the object, the method name gets pasted in for you as soon as you’ve type “of_Get.”

AutoScript now includes Statement Templates. If you select this option in the AutoScript options dialog, then statements like IF THEN ELSE, CHOOSE CASE, FOR NEXT, TRY CATCH FINALLY, and DO WHILE are available in the popup. “TRY CATCH FINALLY”? you say… You’ll have to read on.

Yet another new nicety is the inclusion of meaningful icons that precede the entries in the AutoScript popup. Also new is the option to have No Context entries selected. This means that if you are in an empty script or on a blank line of code, activating AutoScript will display only what you’ve specified in No Context. Include in the AutoScript dialog. This is handy because sometimes you want AutoScript’s assistance right when you start coding, but know that you’ll never start with, let’s say, a method but rather an instance variable.

New Functions

Some new File System functions have been added in PowerBuilder 8, which reduces needs for API calls. File System Function now include ChangeDirectory, CreateDirectory, DirectoryExists, FileCopy, FileMove, GetCurrentDirectory and RemoveDirectory. Having functions like these available in native PowerBuilder will greatly strengthen your applications while slowly but surely laying the PFC to rest.

New Print functions include PrintGetPrinter and PrintGetPrinters. These allow you to attain the current printer name or get a list of available printers. Also new in 8.0 are the PrintSetPrinter and PrintSetupPrinter. Again, the need for API calls are diminishing as well as the need for the PFC.

There have been some new Math and String functions added. Some are both PowerScript and DataWindow Expression functions. New math functions include ASin (calculates the arcsine of an angle), ACos (calculates the arc-cosine of an angle), and ATan (calculates the arctangent of an angle). New string functions include LastPos (finds the last position of a target string in a source string), LenW (reports the length of a string as the number of characters it contains), and WordCap (capitalizes the first letter of each word in a passed script. It sets the remaining letters in each word to lowercase).

Some miscellaneous functions have been added as well. These include ChooseColor (displays the standard color selection dialog box), GetFolder (Displays a folder selection dialog box), as well as a few more.

Improved Error Handling via New Objects

There are some new objects in 8.0, most of which won’t discuss. There are three that I felt were imperative to tell you about. These are the Exception, RuntimeException and Throwable objects. The Exception and RuntimeException objects are both inherited from Throwable. Thanks to the addition of these new objects, Powerbuilder no longer has to fire a single error-handling event to cope with runtime and system errors. You now can use more context-sensitive objects to handle specific errors throughout your applications.

You’ll probably notice the change in the function prototype box when coding your methods. There is a new entry area called Throws. Here you would enter one or many of your own Exception objects. You can either type them into the box separating them with commas or you can select several and drag them into the box. Also, the RuntimeException object is automatically added for you, and as the name implies, it handles runtime errors. Something to note is that when viewing the function signature in the System Tree or Browser, Throws is not displayed.

Closely related to these objects in the new TRY CATCH FINALLY syntax available now in 8.0. I’m not going to discuss these any further as they will be covered in detail by another author.
Wizards and Tools

The wizards used to create objects have improved somewhat. More accurately, PB has widened out in what is accomplished via the wizards. Obviously, you’ve got new items to create such as Workspaces and Targets. Then you also have your standard EAServer objects, components, and fun stuff. You’ve got wizards for creating Web sites, applications, and adding (migrating) existing applications and/or Powersite projects into your Workspace as Targets.

Under the Tools menu, we now have an EAServer Profile item. This is akin to the DataBase Profile we’re used to, only it holds connection information for your various Jaguar servers. This is nice because you don’t have to run script or projects to see if your connections will work. Once you’ve got your profiles in place, they’re available for use throughout your scripting and wizards. Again, I need to leave the details for another author.

Don’t Miss This Ship

In the past, there have been new versions of PowerBuilder released that offered little to no improvements in comparison to its predecessor. A lot of shops decided to not upgrade because the gains were not worth the efforts. Not so with version 8.0. If ever there were a time to upgrade, it’s now. PowerBuilder (EASTudio) is evolving very quickly into an all-encompassing solutions package. If you want to start deploying future-proof, dynamic, Web-enabled solutions while fully capitalizing on your current PowerBuilder skills, then by all means, start using PowerBuilder 8.0 as soon as possible. The changes Sybase has slated for 9.0 will be much easier to adapt to if you’ve already mastered 8.0. It won’t be like skipping from 4.0 to 6.5, or even 5.0 to 7.0, like so many shops did.

Sybase has gotten back on target (no pun intended) with version 8.0. I honestly see this version helping Sybase reclaim some of the ground they’ve lost. By jumping onboard and deploying full-blown solutions for your clients using this release, you’ll be fulfilling two important goals. First, you’ll be making yourself more marketable by being able to build the best client-server and/or Web applications around, all without requiring massive reeducation. Second, you’ll be ensuring PowerBuilder’s place in the future while doing the same for yourself.

This is the best version to date. I highly recommend you start using it as soon as possible. If you’re attending Techwave, sign up for the Saturday classes or attend the 8.0 sessions. Try to get an evaluation disk into your shop to prove how valuable this upgrade will be. By all means, make 8.0 certification your goal. Trust me, this one will be big. You don’t want to miss this ship. More importantly, you don’t want to miss the fun. This version really is enjoyable to use. You can do anything with it.

I have limited time to reply to e-mails and may not get to answer you all, but feel free to write me with your questions, comments and ideas. I hope to see you at Techwave this August! ▼

Author Bio

Tom Peters is director of online applications for Sherwood International in Armonk, NY, an ISV that specializes in insurance software. Tom is a CPD professional with over 15 years’ experience.

tom.peters@sherwoodinternational.com
PowerBuilder 8 is out now. And you’re a developer using PB 7 (perhaps still using PB 6.5 or an even earlier version) and spending the majority of your time on “traditional” client/server applications.

You’re asking yourself, “Does PB 8 have anything for me?” The big new features in PB 7 (Jaguar integration, HTMDW) were not something you could use, and you have about the same enthusiasm about PowerSite integration.

Well, the good news is that there is a slew of new features in PB 8 for “the rest of us.” Stuff that will make our day-to-day development experiences a bit easier.

IDE Changes

WORKSPACES

Ever need to work on more than one application at the same time? Ever get tired of having to switch back and forth between them? Problem solved. PB 8 introduces workspaces (see Figure 1) that are collections of PowerBuilder (and PowerSite) applications. You can open up any of the objects from any of the applications.

SYSTEM TREE

If you’re like me, the one painter you have open all the time is the library painter, which you use to navigate to the library and then the next object you’ll be working on. The only hassle is when the libraries are in different
directories, perhaps even on different drives, and you have to keep navigating across them. Well, no more. The System Tree (see Figure 1) presents all the objects in all the targets so that they're readily accessible. What's more, the System Tree also acts like the object browser by displaying the methods and properties of the objects and allowing them to be dragged-and-dropped into the script editor.

**SOURCE EDITOR**

I don't know how many times a question has come up on the newsgroups that requires accessing object properties that the PowerBuilder IDE doesn't expose, such as using the DataWindow's processing attribute to change a DataWindow from grid to tabular format without reconstructing it from scratch. Generally, the response has been: “export the object, edit the source, and then import the modified code.” At the same time, somewhere else on the planet, a Sybase tech support engineer must give an involuntary flinch.

Well, after seven versions, Sybase has finally decided that they're not going to convince us to stop juggling knives, so the best thing they can do is at least give us sharper ones to work with. The “Edit Source” option in the System Tree (see Figure 3) makes the object source code directly available for editing within the IDE, and the object is immediately updated when the changes are saved.

**Project Compiling**

PB 8 provides a few new features in compiling projects that used to be available only in third-party products, most notably E. Crane Computing's PowerGen and VersionEdit utilities. The first is that the project object itself allows the developer to embed resource information (company name, product name, copyright, version, etc.), which is then compiled into the EXE and DLLs but not PBDs (see Figure 4). They can then be seen, for example, in the Version tab of the Properties dialog when the EXE or DLL is viewed in Windows Explorer. The second, less obvious, enhancement is that project compilation can be invoked through command-line arguments to PowerBuilder, allowing for automation of the compilation process.

This raises the question of the usefulness of those third-party products now that Sybase has incorporated some of that functionality into PB 8. While that depends on just how much of that additional functionality you need, my estimation is that there's still a significant role for those third-party products. For example, the PB 8 command-line compilation process doesn't allow you to pass the version number to be assigned during compilation as E. Crane's VersionEdit does. If you need to incorporate something, such as a daily build number, into the version information, the PB 8 method still doesn't quite meet your needs.

PB 8 also stamps the file with its own values for file version and internal name, properties that products such as VersionEdit provide access to (see Figure 5). Perhaps more significantly, I'm a firm believer in optimizing all the libraries after the full regeneration and before generating the EXE and DLLs or PBDs. PB 8 still doesn't incorporate any ability to automate library optimization as part of the compilation process the way a product such as PowerGen does.

Further, the PB 8 compilation process still leaves the compiled files in the same directories as the source code libraries. That's fine if all the source code libraries are in the same directory.
But if you're working with a set of PFC libraries in one directory, a corporate extension layer in another, some application-specific libraries in another, you'll be doing a lot of moving of files afterward. Products such as PowerGen allow you to specify the directory that all the compiled libraries are moved to after compilation. PowerGen has a number of additional features (such as the ability to compile multiple applications in a batch and to specify which libraries to leave out of the regen/compilation process) that continue to make it a tool that should be in every PowerBuilder developer's toolkit, even with PB 8.

**New PowerScript Functions**

You no longer need to resort to calling Windows API functions or perhaps even custom-written DLLs to handle file and directory operations. PB 8 adds a `FileCopy` function for copying files, and the `.CreateDirectory`, `ChangeDirectory`, `DirectoryExists`, `GetCurrentDirectory`, and `RemoveDirectory` functions for working with directories. Of those, the `ChangeDirectory` function is probably the one I've seen the most requests for, as it's the only way to control the default directory for PowerBuilder's `GetFileOpenName` and `GetFileSaveName` functions. Perhaps PB 9 will give us the full 32-bit functionality of those calls, including specifying the default directory and – for `GetFileOpenName` – selecting multiple files. Until then, I have a custom-written DLL on the www.pfguide.com site or the www.teamsybase.com site to support that.

**Printer Functions**

One long-standing bone of contention for PowerBuilder developers is that PowerBuilder prints only to the default printer. As a result, the developer must resort to third-party utilities such as PowerPrinter or PowerWhizz to automate the switching of the default printer whenever they need to send a print job to a specific printer. No more. PB 8 introduces the `PrintGetPrinter` (get name of default printer), `PrintSetPrinter` (set default printer), and `PrintGetPrinters` (get the list of printers on the client) functions. It also adds the `PrintSetupPrinter` function to display the printer setup dialog to the user. One side effect of this is that Sybase no longer needs to provide the PFC-COM32.DLL file with PFC as a result. The only thing that DLL did was support the printer setup dialog call, so the PFC code has been modified to call the new PowerScript function.

What's even more significant, changing the default printer is no longer the only, and certainly not the best, way to handle this anymore. The printer property for the DataWindow has also been enhanced so that it's no longer read-only. Therefore, when a developer needs to send a particular print job (assuming it's a DataWindow) they need only point the DataWindow printer property to the target printer and leave the default printer alone.

As before, this raises the question of the continued use of various third-party products that provide similar functionality. And again, that depends on what specifically you were using those third-party products for. If all you needed them for was to switch the default printer to reroute certain print jobs, then PB 8 has essentially eliminated your need to rely on them. Those products, however, allow for a great deal of interaction with the client printer. Therefore, if you need to do more than simply change printers, those products will continue to be quite useful.

**Miscellaneous**

Certainly worth at least a mention are some other new PowerScript functions: `LastPos` (like `Pos`, except it starts with the end of the string), `ASin`, `ACos`, `ATan` (for you math buffs), `Sleep` (causes PB to pause without chewing up processor cycles), and `ChooseColor` (which displays the ChooseColor dialog). It's also worth mentioning that the `ShowHelp` function has been updated to support compressed HTML format help files.

**Summary**

That's not all of the new functionality for us "traditional" client/server developers. There are others such as the center property for windows (causes a window to automatically open up centered). For those of you, like myself, who work a lot with Oracle databases, you should be pleased to see that there is now full support in the editors and the database painter for functions and procedures within packages. There are still other features such as the new exception handling keywords, the autoscript enhancements, and the new approach to the source control interface that I just don't have the time to cover in this article. All in all, though, I think you'll be pleased with what you find in PB8.

As I noted in my editorial in this issue of PBDJ, one of the reasons developers should have a lot to be thankful for in this version is that we got the opportunity early in the development cycle to provide feedback on what was important to us. Let's make sure we do the same with PB 9. Get those enhancement requests in. You can use CaseXpress on the support section of MySybase to do that even if you don't have a support agreement.

If Sybase gives us the opportunity again to vote on those enhancements, get in there and vote. I'm going to do my best to make sure that PDF gets added as a SaveAs format, and I hope you'll join me in that.

In the meantime, be sure to stop by "The Sybase TechWave Technology and Solutions Boardwalk" at TechWave 2001. TeamSybase members, including yours truly, will be there to answer your technical questions or just chat. See you then! ▼

**Author Bio**

Bruce Armstrong is a senior PowerBuilder developer with the Los Angeles office of Kforce.com. A member of TeamSybase, he has been using PowerBuilder since version 1.0B. He was a contributing author of PowerBuilder 4.0 Secrets of the Masters.
5 years worth of...
Features & Articles
Product Reviews
Case Studies
Tips & Tricks
Interviews
Editorials
IMHOs
& more!

Check out over 500 articles covering topics such as...
Java Fundamentals, Advanced Java, Object Orientation, Java Applets, AWT, Swing, Threads, JavaBeans, Java & Databases, Security, Client/Server, Java Servlets, Server Side, Enterprise Java, Java Native Interface, CORBA, libraries, Embedded Java, XML, Wireless, IDEs, using Java with others, and much more!

Questions? E-mail JDJCDSYS-CON.COM

www.javadevelopersjournal.com
I realize that many people are talking about nothing but Java, and yes, I’m writing Java components myself; however, PowerBuilder still lives on.

There are many existing PowerBuilder client/server applications in production that need to be maintained and many that need to be moved to the Web.

PowerBuilder 8 has some nice development features that make it an environment for PowerBuilder applications as well as Web applications. This article discusses the new user interface and concentrates on Web targets.

**PowerBuilder 8 User Interface**

There’s a list of new features for this release of PowerBuilder. The first one I’ll discuss is the revamped GUI. Yes, the interface was completely changed in version 7, but it was for the better. It changed again with this release, also for the better. What Sybase has done is integrate the PowerSite concept of workspaces and targets into PowerBuilder.

I also want to tell you that I’m pleased with the integration that Sybase has done. The interface is, in my experience, faster than PowerBuilder 7, and more stable. I know this opens up the floodgates for e-mails, but these types of discussions are what make a product better.

Now to get back to the specifics, the workspace concept is an area that contains multiple targets. You can open only one workspace at a time, but each workspace can open many different types of targets. The targets currently supported are PowerBuilder and Web applications. PowerBuilder applications are those you’re familiar with – client/server and EAServer components. Web applications include Web resources, such as HTML files, scripts, images, and downloaded components. You can now work on multiple PowerBuilder applications at the same time without changing applications. In addition, you can open multiple Web applications. Figure 1 shows a workspace with two PowerBuilder applications and one Web application opened. This allows you to edit HTML pages at the same time as client/server script and any other type of PowerBuilder objects.

![Figure 1: Workspace with multiple targets](image1.png)

The concept of "Recent Objects" is still implemented, as is "Recent Workspaces," which allows for fast switching between objects. Both these options can be set to store the maximum number of occurrences in the System Options window, and the maximum for each is 36. The options also allow for various workspace settings, such as loading the previous workspace when PowerBuilder is started, as shown in Figure 2.

**Web Targets**

A Web target is a Web application that contains all the resources necessary to build a Web site. These resources include HTML files, scripts, and images. Each target also contains information for deployment,
database connections, and build options. Creating a Web target or a resource for a Web target is simple due to the wizards provided. There are two wizards that create Web targets (see Figure 3). The first creates a Web site target only with no configuration information; it must all be added manually.

The second wizard creates a PowerDynamo Web site, identified on the Target tab as a Dynamo Web site. This wizard takes you through many screens, asking all the necessary questions to create an ODBC datasource and a PowerDynamo Web site that’s stored in an Adaptive Server Anywhere (ASA) database. Prior to version 8 this process had to be done from Sybase Central, so this allows it all to be done from one development environment. You can still edit settings from Sybase Central, however, it makes more sense to do everything from a single tool. One thing that needs to be added is for the wizard to create Dynamic File Sites, which is my preference when building Web applications. I’ve requested this feature so hopefully it’ll be available in a maintenance release. You can create a deployment configuration to a Dynamic File Site, however, the site itself needs to be created outside of PowerBuilder. This is not a big limitation but one that I wanted to address.

I won’t go through and review all the screens of the wizard but will leave that for training classes. What I will do is review the Web resources that can be created so that you have a thorough understanding of what to expect when you install PowerBuilder 8. There are seven types of resources that can be created for a Web target (see Figure 4), and I’ll discuss several of them.

The first is a “Quick Web Page,” which does exactly what the name implies – creates a simple HTML page. This page contains basic HTML tags, which are the beginning of a page. This is for those developers who are very comfortable with HTML and want to edit it manually or with the System Tree.

**QUICK WEB PAGE CODE**

```
<HTML>
<HEAD>
    <TITLE></TITLE>
</HEAD>
<BODY PSPARAMS="">
</BODY>
</HTML>
```

The second Web resource is a “Web Page” that takes you through various screens that are prompting for Web Page information so the page can be created. This creates an HTML page with the tags indicating what was specified. The following list summarizes the information that’s asked with the wizard screens.

- Page title
- File name
- Stylesheet
- Background image
- Background color
- Is the header based on the title?
- Place the date created in the footer

The code in Listing 1 was generated when all the options were selected with the wizards.

The third Web resource is a 4GL Web Page. This is a feature that I’ll discuss in detail in my next article. The 4GL Web Page is an extension of a standard Web page that incorporates an event model to enable the creation of dynamic content. The event model takes care of many implementation details for database connections and EAServer component access so that the developer can concentrate on the Web pages and business logic. This is truly one of the significant features of PowerBuilder 8 and one that will allow developers to learn more quickly how to interface with EAServer components.

The fourth Web resource is the Web DataWindow Page, formerly known as the Design Time Control. This wizard takes you through the process of specifying the database connection information, PBL name, dataobject name, and other pertinent information about the DataWindow. It then creates the PowerDynamo code to generate a Web DataWindow as HTML and JavaScript so it can be displayed in a browser. The new HTMLGenerator80 component offers many new component methods that make the Web DataWindow even more robust than the version 7 implementation. Again, this is another topic to be covered in a future article.

The fifth Web resource is the FrameSet Page, which is a wizard that will create HTML frame syntax based on the selected style. It provides six different styles to choose from to quickly create the frames shown in Figure 5. Once the code has been created it can easily be edited either with the page properties dialog or by editing the code in the source tab. This is something that’s very useful when learning HTML and how to set up frames.

**FIGURE 2** Workplace options  
**FIGURE 3** Targets  
**FIGURE 4** Web resources  
**FIGURE 5** frameSet types  
**FIGURE 6** Stylesheet editor

Continued on page 19
Toward whom is this release oriented?

This release was designed with the needs of our customer base in mind. Many of our customers plan to maintain and develop existing and new two-tier applications with PowerBuilder 8. For these customers, we've added many features that help build their applications faster and more efficiently. For customers who need to bring their existing applications to the Web or distributed environments, we've created new applications for these architectures. PowerBuilder 8 offers new capabilities within a familiar IDE.

What specific productivity capabilities are contained in the new release?

There are many changes in the IDE, such as the Clip Window, which stores frequently used code snippets, and the Output Window, which is used to show results of migrations, builds, deployments, project execution, object saves, and searches. There's also the System Tree that displays information about your applications, objects, syntax, and object models for client- and server-side scripting in Web applications and components in EAServer. It also provides easy-to-use drag-and-drop programming. New Java-style exception handling gracefully handles errors where they occur in an application through better object-oriented design methodology.

What new features enable or ease the development of Web and n-tier applications?

We have some cool new wizards that will guide you through object creation and deployment for next generation applications. You can build and deploy components into EAServer without leaving the PowerBuilder IDE. For Web development, the DynamoWizard will build an ASA database and create the Web site in the database. It will also create the target as well as the build, debug, and deployment configurations. The Web Target Object Model adds a layer of abstraction that makes building and deploying Web applications a truly RAD experience.

Anything else new about the integration with EAServer?

PowerBuilder developers can be immediately productive using EAServer out of the box. We've tightly integrated PowerBuilder with EAServer in several ways. By using Wizards, you can easily build and deploy components into EAServer without leaving PowerBuilder. PowerBuilder components can now be built based on existing interfaces inside of EAServer – this means that your design group can focus on design, not on how the applications in your organization will be written.

New exception handling capabilities enable PowerBuilder NVOs residing in EAServer to throw user-defined exceptions and gracefully handle other exceptions generated within intercomponent calls, regardless of the other component’s implementation. Similarly, PowerBuilder clients can handle exceptions thrown from PowerBuilder, Java, C/C++, and EJB components providing a much more robust and maintainable end-user application.

What are the benefits of using PB to build Web and n-tier applications over Java?

With Java, developers face two learning curves – understanding the new architectures that they’re building for as well as learning the Java language. With PowerBuilder, one learning curve is removed from the picture and developers will be able to immediately contribute to their company’s Web, n-tier, and e-business development efforts using a familiar tool. The true cost of moving to Java is often underestimated by companies. It’s a significant effort to learn the language well enough to develop sophisticated applications that PowerBuilder developers are accustomed to building and delivering quickly. Since Java is a 3GL, it’s more difficult and time-intensive by definition. The 4GL RAD power of PowerBuilder enables developers to quickly deliver next-generation applications.

Can you provide us with any insight into the features and timing of the next release of PowerBuilder?

We’re in the planning stages now for the next release of PowerBuilder, and we’re looking for input from PowerBuilder users. Currently, we plan to continue to strengthen our integration with EAServer and implement new functionality provided by that product. We have some other great ideas for the next release, and we’re continuing to look for requirements and features customers would like to see in the product.
The sixth Web resource is the Cascading Style-sheet, which is a wizard that creates an empty stylesheet that can be edited from PowerBuilder (see Figure 6). Stylesheets offer a wide variety of settings for many things within a Web page. The editor allows you to specify the tags that are supported and then specify the settings that exist within these tags. One drawback here is that you need to understand how CSS works, then you can use the editor.

**Summary**

This article covered the new interface in PowerBuilder 8 and the workspaces and targets that are supported. As with PowerBuilder 7, this new interface will improve the productivity of developers, something that management is always interested in.

We discussed how Web targets work and what they’re used for, such as making it easy to use PowerBuilder to create client/server applications, EAServer components, and Web sites all from the same development environment. Finally, no need to learn multiple tools just to create a PowerBuilder application and Web site.

**Author Bio**

Larry Cermak is president of Branick Consulting Inc., a consulting firm specializing in the Sybase family of products. He has been working with the Web DataWindow since it was first introduced. He’s a member of Team Sybase, a writer for the Sybase Developer Network and PBDJ, a frequent speaker at Sybase conferences and seminars across the country, and is author of *The Web DataWindow*.

**Listing 1**

WebPage Code

```
<HTML>
<HEAD>
<TITLE>WebPage Title</TITLE>

<Link href="menu.css" rel="stylesheet type=text/css">
</HEAD>
<BODY bgcolor="Silver" background="branick_banner.gif">
PSPARAMS="">
<Hi>WebPage Title</Hi>
Put your data here
</P>Created on 04/11/2001</P>
</BODY>
</HTML>
```
When I first learned Java, I was amazed at how much it resembled PowerBuilder. The two languages are strikingly identical in how they implement inheritance and polymorphism. Over the past month, I have previewed PowerBuilder 8. Guess what? PowerBuilder is starting to look like Java.

What first caught my eye is how PowerBuilder 8 handles runtime errors. Before I discuss the new features, I would first like to cover how previous versions of PowerBuilder handle runtime errors. Let’s begin with a history lesson.

Error Event

This event is fired if a runtime error occurs on a Connection, JaguarORB, DataWindow, or OLE control. Since this event gives us many arguments to work with, these arguments tell us the nature and the severity of the error. Code is placed in this event to handle the error.

If the error is not severe, perhaps we ignore it. Maybe we let the user off the hook with a message box – a simple slap on the wrist. If no code exists in this event, then the error structure information is used to populate the global error variable and the SystemError event on the Application object is triggered.

SystemError Event

In the SystemError, event code can be placed to handle the runtime error. As in the Error event, we can choose what we want to do. If no code is placed in the SystemError event, the application goes into crash-and-burn status.

What’s the big deal? Well, the problem with the “old way” is that error handling occurs far from the source. This can really limit how we can handle runtime errors. PowerBuilder 8 changes all this with “exception” handling. Exception handling allows programmers to deal with runtime errors, right at the source.

What’s an Exception?

The term exception is shorthand for the phrase “exceptional event.” It can be defined as follows: An exception is an event that occurs during the execution of a program that disrupts the normal flow of instructions. Many kinds of errors can cause exceptions – problems ranging from serious hardware errors, such as a hard disk crash, to simple programming errors, such as trying to access an out-of-bounds array element.

When such an error occurs within a PowerBuilder script, PowerBuilder creates an exception object and hands it off to the runtime system.

The exception object contains information about the exception, including its type and the state of the program when the error occurred. The runtime system is then responsible for finding some code to handle the error. In new terminology, creating an exception
object and handing it to the runtime system is called throwing an exception.

After a method throws an exception, the runtime system leaps into action to find someone to handle the exception. The set of possible “someones” to handle the exception is the set of methods in the call stack of the method where the error occurred. The runtime system searches backward through the call stack, beginning with the method in which the error occurred, until it finds a method that contains an appropriate exception handler.

An exception handler is considered appropriate if the type of exception thrown is the same as the type of exception handled by the handler. Thus the exception bubbles up through the call stack until an appropriate handler is found and one of the calling methods handles the exception. The exception handler chosen is said to catch exception.

The interesting premise of this technology is that Java handles runtime errors (now called exceptions) the same way.

By using exceptions to manage errors, PowerBuilder has new advantages over traditional error management techniques:

• Advantage 1: Separating Error Handling code from “regular” code
• Advantage 2: Propagating errors up the call stack
• Advantage 3: Grouping error types and error differentiation

**Advantage 1: Separating Error Handling Code from “Regular” Code**

In traditional PowerScript, error detection, reporting, and handling often lead to confusing spaghetti code. For example, suppose you have a function that reads an entire file into memory. In pseudo-code, your function might look something like this:

```java
of_readfile(
  open the file;
  determine its size;
  allocate that much memory;
  read the file into memory;
  close the file;
)
```

At first glance, this function seems simple enough, but it ignores all of these potential errors:

• What happens if the file can’t be opened?
• What happens if the length of the file can’t be determined?
• What happens if enough memory can’t be allocated?
• What happens if the read fails?
• What happens if the file can’t be closed?

To answer these questions within your function, you’d have to add a lot of code to do error detection, reporting, and handling. Your function would end up looking something like Listing 1.

With error detection built in, your original seven lines have been inflated to 24 lines of code – a bloat factor of almost 400%. Worse, there’s so much error detection, reporting, and returning that the original seven lines of code are lost in the clutter. And, worse yet, the logical flow of the code has also been lost in the clutter, making it difficult to tell if the code is doing the right thing. (Is the file really being closed if the function fails to allocate enough memory?)

It’s even more difficult to ensure that the code continues to do the right thing after you modify the function three months after writing it. Many programmers “solve” this problem by simply ignoring it – errors are “reported” when their programs crash.

Now we have an elegant solution to the problem of error management: exceptions. Exceptions enable you to write the main flow of your code and deal with the, well, exceptional cases elsewhere. If your function used exceptions instead of traditional error management techniques, the pseudo-code would look something like that in Listing 2.

Note that exceptions don’t spare you the effort of doing the work of detecting, reporting, and handling errors. What exceptions do provide is the means to separate all the grungy details of what to do when something out-of-the-ordinary happens from the main logic of your program. In addition, the bloat factor for error management code in this program is about 250% – compared to 400% in the previous example.

**Advantage 2: Propagating Errors Up the Call Stack**

A second advantage of exceptions is the ability to propagate error reporting up the call stack of methods. Suppose that the readFile method is the fourth method in a series of nested method calls made by your main program: method1 calls method2, which calls method3, which finally calls readFile.

```java
method1 {
  call method2;
}
```

Suppose also that method1 is the only method interested in the errors that occur within readFile. Traditional error notification techniques force method2 and method3 to propagate the error codes returned by readFile up the call stack until the error codes finally reach method1 – the only method that is interested in them (see Listing 3).

PowerBuilder searches backward through the call stack to find any methods that are interested in handling a particular exception. A function can “duck” any exceptions thrown within it, thereby allowing a method farther up the call stack to catch it. Thus only the methods that care about errors have to worry about detecting errors.

```java
method1 {
  try
    call method2;
  catch (exception) {
    doErrorProcessing;
  }
}
```

However, as you can see from the pseudo-code, ducking an exception does require some effort on the part of the “middleman” methods. Any checked exceptions that can be thrown within a method are part of that method’s public programming interface and must be specified in the throws clause of the method. Thus a method informs its callers about the exceptions that it can throw, so that the callers can intelligently and consciously decide what to do about those exceptions. Note again the difference in the bloat factor and code obfuscation factor of these two error management techniques. The code that uses exceptions is more compact and easier to understand.

**Advantage 3: Grouping Error Types and Error Differentiation**

Often exceptions fall into categories or groups. For example, you could imagine a group of exceptions, each of which
The interesting premise of this technology is that Java handles runtime errors (now called exceptions) the same way.

Furthermore, you can imagine that some methods would like to handle all exceptions that fall within a category (all array exceptions), and other methods would like to handle specific exceptions (like the null object references, please).

Because all exceptions that are thrown within PowerBuilder are first-class objects, grouping or categorization of exceptions is a natural outcome of the class hierarchy. PowerBuilder exceptions must be instances of Throwable or any Throwable descendant.

The PowerBuilder class Exception is also derived from Throwable and is typically used as an ancestor object for user-defined exception types. It is the root class for all checked exceptions. In addition to system exceptions, you can create your own. User-defined exception handling is beyond the scope of this article. I will discuss user-defined exceptions next month.

What's Next?

Now that you understand what exceptions are and the advantages of using exceptions in your PowerBuilder programs, it's time to learn how to use them.

Try-Catch

When trapping runtime errors within PowerScript use the Try-Catch block. Before I show you how this block works, it will be a good idea to show you some pre-PowerBuilder 8 code that doesn't have any exception handling. See the code example in Listing 4.

The above code has at least three major problems. First of all, while we are performing division, what happens if we divide by zero? Furthermore, what happens if the window w_window is not currently open? Finally, the last line of code is trying to access an array element that does not exist. Does this code compile? You betcha. But there are at least three runtime errors that can occur. In this legacy piece of code, error handling would have to take place in the Application object's SystemError event. But not anymore.

These runtime errors can be caught in a TRY-CATCH. As you will see the syntax is straightforward and easy to use. In PowerBuilder 8, the code would look like that in Listing 5.

The PowerBuilder class Exception is straightforward and easy to use. In PowerBuilder 8, the code would look like that in Listing 5.

This is the simplest syntax of a TRY-CATCH. It works exactly like an IF statement. The code in the TRY is treated specially. If a runtime error occurs, PowerBuilder notes the type of runtime error that has occurred and looks to see if you “caught” it. If so, the code within the CATCH block is fired. Does this make sense?

In the last portion of the code in Listing 5, we are catching a RuntimeException (which is the generic way of catching ALL runtime errors). Since a runtime error occurred in our TRY block – rather than the user crashing and burning – the user is shown a friendly message and we all can breathe a sigh of relief. Can multiple lines exist in a TRY-CATCH? Yes. Take the following code example:

```java
try
  st_result_text = String(li_result)
  w_window.visible = FALSE
messagebox("Error",e3.getMessage())
end try
```

In this example, if any runtime error occurs within the TRY block, the code in the CATCH is fired. Here we are displaying the nature of the system error via the getMessage method on the RuntimeException class. What can you do with an exception once you’ve caught it? It’s really up to you.

• Fix the problem and try again.
• Do something else instead.
• Exit the application with HALT CLOSE.
• Rethrow the exception.
• Throw a new exception.
• Return a default value.

Catching Multiple Exceptions

In the above example, we have a TRY-CATCH block for each individual operation we wanted to do. It worked just fine, but there is even a better way. You may be wondering if you are allowed to catch multiple exceptions. Well, you can. The example in Listing 6 catches multiple exceptions.

In this code, if multiple blocks match the exception type, the first block that matches the type of the exception catches it.

Next Month

This should get you going with exception handling. Since there is so much more to talk about, next month I’ll give you the lowdown on advanced exception-handling techniques as well as how to write your own exception classes.

---

**Author Bio**

Bob Hendry is a PowerBuilder instructor for Envision Software Systems and a frequent speaker at both national and international PowerBuilder conferences. He specializes in PFC development and has written two books on the subject, including Programming with the PFC 6.0.
```plaintext
Listing 1

of_readfile()
initialize errorCode = 0;
open the file;
if (theFileIsOpen)
determine the length of the file;
if (gotTheFileLength)
allocate that much memory;
if (gotEnoughMemory)
read the file into memory;
if (readFailed)
errorCode = -1
else
errorCode = -2
else
errorCode = -3
close the file;
if (theFileDidntClose and errorCode = 0)
errorCode = -4
else
errorCode = errorCode and -4
else
errorCode = -5
end if
return errorCode

Listing 2

of_readfile()
try
open the file;
determine its size;
allocate that much memory;
read the file into memory;
close the file;
catch (fileOpenFailed)
doSomething;
catch (sizeDeterminationFailed)
doSomething;
catch (memoryAllocationFailed)
doSomething;
catch (readFailed)
doSomething;
catch (fileCloseFailed)
doSomething;
end try

Listing 3

method1 (errorCodeType error;
    error = call method2;
    if (error)
        doErrorProcessing;
    else
        proceed;
    }
errorCodeType method2 (errorCodeType error;
    error = call method3;
    if (error)
        return error;
    else
        proceed;
    }
errorCodeType method3 (errorCodeType error;
    error = call readFile;
    if (error)
        return error;
    else
        proceed;
    )

Listing 4

// Local Variables
Int li_first
Int li_second
Int li_result
String ls_names[10]

// Do some division
li_first = Integer(sle_first.Text);
li_second = Integer(sle_second.Text);
li_result = li_first / li_second
st_result.text = String(li_result)

// Make 'other' window invisible
w_window.visible = FALSE

// Add Names

Listing 5

// Local Variables
Int li_first
Int li_second
Int li_result
String ls_names[10]

// Do some division
li_first = Integer(sle_first.Text);
li_second = Integer(sle_second.Text);
li_result = li_first / li_second
try
st_result.text = String(li_result)
catch(DivideByZeroError e1)
    messagebox("Error","Cannot divide by Zero")
    return
end try
// Make 'other' window invisible
try
w_window.visible = FALSE
    catch(NullObjectError e2)
    messagebox("Error","Window does not exist")
    return
end try
// Add Names
try
    catch(RuntimeError e3)
        messagebox("Error","Array element does not exist")
    return
end try
// Do some division
li_result = li_first / li_second
    w_window.visible = FALSE
    catch(DivideByZeroError e1)
        messagebox("Error","Cannot divide by Zero")
    catch(NullObjectError e2)
        messagebox("Error","Window does not exist")
    catch(RuntimeError e3)
        messagebox("Error",e3.getMessage())
end try
```

The code listing for this article can also be located at www.PowerBuilderJournal.com
Plan to Attend...This 4-Day Cutting-Edge Conference

XML...Dramatically Changing the Substance of Corporate Systems

- XML will fundamentally improve the speed, cost and flexibility of business applications.
- IT Managers expect XML Technology to enable performance of new operations and shorten application development time by 30% to 50%.
- E-commerce transactions using XML are predicted to rise from .5% in 2000 to more than 40% by the end of 2003.
- XML products now under development will hit the market in the second half of 2001. See them first at XMLEdge!

www.XMLEdge2001.com

Conference Tracks

Track 1: PRESENTING XML
The latest tips, tricks and strategies for presenting XML in various front-end presentation environments
XSL / XSLT / CSS / XLINK / XPOINTERS
XHTML
BROWSER SUPPORT FOR XML APPLICATIONS
WML AND WAP
VOICEXML
SCRIPTING AND XML (PERL / PHP / PYTHON / JAVA SCRIPT)
WEB SERVICES
WIRELESS XML: WAP / WML, XML AND MOBILE COMMUNICATIONS

Track 2: REAL-WORLD XML
Actual Case Studies using XML and related technologies in various industry sectors
XML TOOLS / UTILITIES APPLIED TO ACTUAL APPLICATIONS
DOCUMENT AND CONTENT MANAGEMENT (EDITORS / PARSERS / SERVERS / TRANSLATORS)
REAL-WORLD APPLICATIONS
XML ISSUES IN THE REAL WORLD PERFORMANCE / SECURITY / COMPRESSION / SERIALIZATION
USE OF XML IN INDUSTRY SECTORS (MANUFACTURING / HIGH TECH, ETC.)

Track 3: XML & JAVA
The latest developments in XML and Java environments as well as insights into developing solutions using XML, Java and related technologies
DOM/SAX / JDOM
JAVA AND XML PROGRAMMING (JSP, IMS, ETC.)
PARSING XML IN JAVA
XML AND DATABASE PROGRAMMING (XQL)

Track 4: EAI with XML
Comprehensive coverage on developing Enterprise Application Integration solutions with XML
XML AND APPLICATION SERVERS
B2B SERVERS
PROCESS AND WORKFLOW INTEGRATION
XML-BASED INTEGRATION USING MESSAGING SYSTEMS
ERP INTEGRATION
DATABASE INTEGRATION

Track 5: XML STANDARDS
The latest news and updates on emerging XML standards from the industry gurus
W3C STANDARDS
XML DESIGN PATTERNS
DOM/SAX / JDOM
ROSSETTANET
SOAP / BIZTALK
DTDS AND XML SCHEMAS
XML REPOSITORIES (BIZTALK / OASIS)
The Largest International XML Conference of the Year! Reserve Your Exhibit Space Now!

Get the Competitive Edge...

- Over 100 exhibitors will have access to the most qualified audience of potential buyers and specifiers.
- They will demonstrate, meet the movers and shakers, solve their challenges and sell!

Plan to Sponsor – Many Sponsorships Are Exclusive!

- Demonstrate your commitment and capabilities to the largest XML audience to assemble on the West Coast in 2001!
- Rise above the noise. Deliver your message with the marketing support of SYS-CON Events and SYS-CON Media.

Who Should Attend XMLEDge2001

- i-Technology Professionals
  - Developers, Programmers, Engineers
- Technical Architects
  - Senior IT / IS Management
- Analysts, Consultants
  - Product & Program Managers

The only XML event backed by the full marketing power of SYS-CON Media and XML-Journal is your best opportunity to reach the largest, most qualified audience of software developers and i-technology professionals in 2001.

Owned and Produced by

135 Chestnut Ridge Road, Montvale, NJ 07645
201 802-3069 • Fax: 201 782-9601
Visit www.XMLEDGE2001.com for more information

Who Should Attend XMLEDge2001

Call 201 802-3069 to reserve your space today!

Last Year’s Exhibitors Include...

- Baltimore Technologies
  - DMU Magazine
  - EarthWeb
- Elad Technologies, Inc.
  - E-Numerate Solutions, Inc.
  - E-Security Journal
  - Fawcette Technical Publications
  - GeekCruises.com
  - Hit Software, Inc.
  - IAM Consulting
  - IBM
  - Infotopia Corporation
  - Ingentix Corporation
  - Insight Software, Inc.
  - Interknack Corporation
  - IXIASOFT, Inc.
  - JavaPro
- LearningPatterns.com
  - Mailers Software
  - Merant
  - Microsoft
  - Mindgap Systems Ltd.
  - MSDN Magazine
  - Nanoby
  - Neocore, LLC
  - Netfish Technologies, Inc.
  - Oasis
  - Object Management Group
  - O’Reilly Network
  - Planet 7 Technologies Corp.
  - Progress Software Corporation
  - Pybix
  - Radiant Logic, Inc.
  - Rogue Wave Software, Inc.
  - SD Times
  - Seagull
- Search XML Resources.com
  - Sequoia Software Corporation
  - Skill Server
  - SoftQuad Software, LTD.
  - Sun Microsystems
  - Talva Corporation
  - The Gilbane Report
  - Tibco Software, Inc.
  - Whitehill Technologies, Inc.
  - X-Aware
  - XML Global Technologies, Inc.
  - XML-Journal
  - XML Magazine
  - XML Times
  - XML.com
  - XML.org
  - XML/EDI Group
  - XMLResources.com
  - XYZ Find


Neither SYS-CON Media nor SYS-CON Events, Inc., is a sponsor of, or is affiliated in any way with, CAMELOT Communications, Inc., or any events produced by CAMELOT Communications, Inc. Java and Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc., in the United States and other countries. SYS-CON Media and SYS-CON Events are independent of Sun Microsystems. The XMLEDGE 2001 International Conference & Expo is no way related to XMLEDGE by Mobile, a platform for a complete wireless internet infrastructure for developing, integrating, managing and evolving applications for the broadening spectrum of wireless devices.
Many PowerBuilder programmers over the last year have been spending most, if not all, of their efforts developing Web-enabled applications or distributed solutions – solutions that take advantage of Sybase’s Enterprise Application Server.

Almost every new project these days can be more portable, more cost-effective and, over time, easier to maintain by using an application server. The new PowerBuilder 8 has some real improvements that help developers get components built faster and deployed more easily.

I will start with the high level of five integration issues addressed in this release, followed by some hands-on “what to do” screen shots to help you dive right in. If you would like more detail on any of these areas, you will find them listed under these topic headings in the Welcome to PowerBuilder 8.0 help menu selection.

• Creating an EAServer component that implements an existing interface
• Client- and component-managed EAServer transactions
• SSL connections and callbacks
• Concurrency property for EAServer components
• Multiple PowerBuilder VMs in EAServer

Implementing an Existing Interface

Now you can easily customize standard API interfaces. When using the wizards, if the “Implement an existing EAServer remote interface” option is taken, the IDL for the existing component will be used to create a custom class user object that contains the methods and properties. Then code your implementation and deploy.

If there is a Java component that your application is using and you think you can write it better in PowerBuilder, simply grab the interface and give it a try!

You can create a PowerBuilder implementation of any existing EAServer component. This option looks like it’s intended mainly for users of Sybase Financial Server because the documentation specifically notes building implementations for the Open Financial Exchange (OFX), Financial Information eXchange (FIX), and Society for Worldwide Interbank Financial Telecommunications (SWIFT) protocols.

There will be other times this option will be useful – as with anything new, experiment a bit with it and see what you can do.

Streamlining Transactions

Remote control of transactions is what this really comes down to. New functions were added that allow code access to more information from the CORBACurrent service object. Clients and components can manage transactions in a server that is running the new two-phase commit transaction coordinator (OTS/XA).
Here are some quick bits on methods from the new functions that help manage transactions. Check the help documentation for more details, syntax, and parameters.

- **GetTransactionName**: Typically used for debugging, GetTransactionName returns a string identifying the transaction associated with the calling thread.
- **RollbackTransaction**: Rolls back the transaction associated with the calling thread.
- **BeginTransaction**: Creates a transaction and associates it with the calling thread by trying to set the transaction context of the calling thread. This call will fail if a thread is already associated with a transaction. If successful, the calling thread can now obtain information about the transaction and control commits and rollbacks.
- **Init**: Has a few quirks you must deal with. First, use the GetContextService to get a reference to the CORBACurrent object.

```plaintext
CORBACurrent i_CORBACurrentobject
GetContextService (CORBACurrent, i_CORBACurrentobject )

Now use the reference i_CORBACurrentobject.init (connection or URL) call where you pass either a connection object or a URL to an EAServer box.

i_CORBACurrentobject.Init( i_connect )
// OR
i_CORBACurrentobject.Init( "iiop://EAServer:9000")
```

By using the connection object the code is portable, so the first option would be preferred. You must perform these initialization steps before using any of the other methods on i_CORBACurrentobject. Also note, if the component is not marked as OTS style the initialization will fail.

- **SetTimeout**: This allows you to specify the number of seconds that can elapse before a transaction is rolled back. This setting applies to transactions created by subsequent invocations of BeginTransaction. The default is 0 or no time out.
- **CommitTransaction**: Completes the transaction associated with the calling thread; however, the transaction will not be completed if any other participants in the transaction vote to roll it back.
- **ResumeTransaction**: Passing this function a Handle to any transaction will associate the calling thread to the transaction. Note that the only way to obtain a valid Handle for the transaction is by using the SuspendTransaction function.
- **SuspendTransaction**: This will return a Handle for the transaction the calling thread is currently associated with. This Handle can be used by any thread in the same execution environment. Calling SuspendTransaction will disassociate the current thread from the transaction.
- **GetStatus**: Using GetStatus will determine what is happening within a transaction that was initiated by a client or component using the BeginTransaction function. The codes returned indicate if the transaction has started, is in the process of preparing, committing, in transition, or is rolling back.
- **RollbackOnly**: Can be used to “vote” which marks the current transaction for rollback.

### SSL Connections and Callbacks

The SSLCallBack Object has been implemented so you no longer have to configure your SSL connections to not require the callback object. You can now provide your own implementation.

Tell EAServer to use your implementation by specifying the name of the object in the callbackImpl SSL property. This global property is set by calling the SetGlobalProperty function.

### Concurrency

Finally, this is an enhancement we have been waiting for. If your standard components do not have this checked, and you're not 100% sure
why – check it. Your Web applications will see a significant performance gain.

According to the PB 8 help files: “The concurrency property determines whether multiple instances of a component can be created to handle multiple client requests. An instance of a PowerBuilder component executes in its own session, and each session can only support one thread of execution. Therefore a single PowerBuilder component instance cannot simultaneously execute multiple client requests. However, multiple instances of the same component can each execute a separate client request.”

Think of it as cloning without all the overhead costs of research, the legal expenses, or ethical questions. When concurrency is checked, clients do not have to stand in line to use your component. Who wants to wait in line anyway?

Previously, this property could be set in EAServer, but by making it a deployment option in PB 8 it will save everybody time.

Multiple PowerBuilder VMs

This will save you some migration headaches: Write once, run forever? Now that would have to be some pretty good code. As long as the PB Virtual Machine is on the server it can be hosted by EAServer 3.6 and later.

Getting Down to Work

Before you try anything, set up your EAServer profile. When I first tried to create components I had a problem. I did not have any EAServer profiles. Save yourself the trouble. Click on this new tool EAServer Profiles selection in PowerBar1 (see Figure 1).

This will pop open the EAServer Profiles window (see Figure 2). I have two set up presently.

Add your EAServer here. If you do not know what the settings should be, talk to your EAServer administrator. Presently the wizards require you to have a profile set up to create components. The creation process will not be allowed to go forward if profiles are not available.

There are two ways to create components: as a Target component or as a component within an existing Target.

Before I bash something, please note: I wrote this from the final beta, so if Sybase dealt with these gripes, great. If not, well, these are the things that annoyed me – and consider this an enhancement request.

Presently, the wizards do not allow adding new profiles on the fly or an option to specify a deployment server at a later time.

In addition, the wizards are “slower than desired” when you have an existing profile set up but the computer currently does not have access to EAServer. This makes it harder for you to write code remotely. I can still develop, but the wait seems a little silly.

Show Me

The new deployment project options look like Figure 3:

• Full rebuild option (general)
• Collapse class hierarchy (general)
• Concurrency (component)
• Support instance pooling (component)
• Specify how many instances to create (component)
• Use existing IDL on EAServer (component)

The complexity required by distributed architecture is a drawback, but the benefits are worth the effort for many applications. PB 8 is shielding developers from these complexities and allowing for better access.

If you need or want to, dive into more complex issues like the two-phase commit transaction coordinator (OTS/XA). This tool allows enough options for developers to accomplish many complex tasks without needing to worry about managing EAServer. I am not 100% satisfied with the 8.0 release, but PB 8 will make me a more productive developer and make managing EAServer boxes easier.
Guaranteed Best Prices
JDJ Store Guarantees the Best Prices.
If you see any of our products listed anywhere at a lower price, we'll match that price and still bring you the same quality service.

Terms of offer:
• Offer good through November 30, 2001
• Only applicable to pricing on current versions of software
• Offer does not apply toward errors in competitors’ printed prices
• Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.

Attention Java Vendors:
To include your product in JDJStore.com, please contact tony@sys-con.com

SHOP ONLINE AT JDJSTORE.COM FOR BEST PRICES OR CALL YOUR ORDER IN AT 1-888-303-JAVA

Guaranteed Lowest Prices!

Guaranteed Best Prices
JDJ Store Guarantees the Best Prices.
If you see any of our products listed anywhere at a lower price, we'll match that price and still bring you the same quality service.

Terms of offer:
• Offer good through November 30, 2001
• Only applicable to pricing on current versions of software
• Offer does not apply toward errors in competitors’ printed prices
• Subject to same terms and conditions

Prices subject to change.
Not responsible for typographical errors.

Attention Java Vendors:
To include your product in JDJStore.com, please contact tony@sys-con.com

SHOP ONLINE AT JDJSTORE.COM FOR BEST PRICES OR CALL YOUR ORDER IN AT 1-888-303-JAVA

Guaranteed Lowest Prices!
Bearing that in mind, it occurred to me that I should begin to test the limits of PowerBuilder 8, and see if I could disprove Muller. It seemed to me the easiest way to go about this heady undertaking was to convert our EMS natural gas trading and accounting application. It was written in PB 6.5 with the PFC and Transact-SQL for the motors behind the screens.

Last year in *PBDJ* (Vol. 7, issue 9), I did a case study of this application and our use of CAST to map the many objects in PowerBuilder and Sybase, and the use of CAST.

EMS has more than 600 PB objects, more than 1,700 stored procedures, 604 triggers, 179 views, and more than 400 tables. Migrating this application was not something that happened quickly, but PB 8 made this a somewhat painless transition.

It occurs to me as I begin to look at my beta version of PB 8 that I’m quite comfortable with PB 6.5 and not so sure that I want to move off it and into the unknown. PB 6.5 is solid, is marketed by Sybase as solid, and was designed as a developer’s environment.

It also flew in the face of the old implementation theory, “Let the other guy work the bugs out.” Instead of ducking, or waiting for the coming storm, I decided to plow ahead....

**Starting the Migration**

The first step in starting this migration was to move all of my pibbles and objects to one of my local drives. This is an important step in speeding up your migration. If you attempt to migrate from a network, it can take a lot of time and tie up a PC for a good while, creating unnecessary network traffic and tying up your development environment—slowing productivity and losing programmer time.

Doing the migration locally and then copying back to the network is a neces-
Deploy HTML, images and other resources needed for a Web site,” to quote Sybase.

Select the Application

Once you have created a Workspace for your applications, select the application that controls the pibbles that you’re going to migrate. This will bring up a list for the migration, and here is where PowerBuilder will stop you with the Migrate Application dialog box and tell you that the application you are opening was from an older version and must be migrated.

It will also give you the opportunity to browse for more pibbles if you have more that you need to add.

Make sure you connect to the database so that when PB starts regenerating the DataWindow objects, it will find the tables and columns in question. As you start the process, PowerBuilder will begin by naming your application object with a .PBT extension, making it into a target – which is the PB 8 way of saying application object. Then it will begin reading in the objects in the pibbles that you are moving, and regenerating and rebuilding them in PB 8.

Now You Can Edit

Now the system will allow you to edit your objects and work in PB 8. Another nice feature of the target was searching by target. It has always been frustrating to search by one pibble at a time, plumbing your way through, so being able to search through a target gives you the ability to run through multiple pibbles in no time at all. Bravo, Sybase!

PowerBuilder uses the Migration Assistant to move through your pibbles, objects, and methods, then scans them, and moves them to the newer version. In the process it captures the errors, obsolete functions, and obsolete events – and then lists them so that you can fix them if necessary.

The Migration Assistant (see Figure 1) can also be used selectively to pick a single pibble or object to migrate. You can choose PFC or PowerScript and PB will check for obsolete functions or events and it will also go through the code for problems with reserved words. The display will show the obsolete code, and the pibble, object name, and line where it’s found along with a suggested replacement. This all comes in for display in the Output window at the bottom of the screen (see Figure 2).

Capturing Errors

PowerBuilder has a nifty new way of capturing the errors that can come up as well. As it began to crash and burn, I got a message saying, essentially, “PowerBuilder is about to crash, do you want to capture the information?” If you get this, you can get a sys dump of sorts, and save it off to disk in a text file format.

This will be very helpful in tracking down some of those mysterious PowerBuilder system deaths that occur from time to time. This text file shows the registers active, the O/S version, the memory status, and what modules are loaded at the moment of pain.

I did get more than my fair share of obsolete objects or renamed ones in my information area of the Migration Assistant. I also had a good number of conflicts between instance variable names and exceptions. Sybase is still not killing these things off summarily, which is both wise and merciful on their part.

Someday dwDescribe may actually stop working, but for now, it just keeps rolling along. PowerBuilder also has several new-reserved words, and you’ll have to change any variable names that conflict with these: TRY, CATCH, FINALLY, THROW, and THROWS.

After I generated all of my PBDs and executables, I had to make up new folders for the PB 8 runtime DLLs. While it’s not something that we’re likely to forget, it does pay to mention it after all. If you follow lists and patterns as I do, it pays to have every detail mentioned.

Creating a Workspace

Once you’ve done that, you’ll have to create a Workspace. This PowerBuilder object gives you a means of grouping applications or targets. This will have a .PWB extension. The Workspace is new to PB 8 and it will give you a way of opening and deploying multiple targets at once.

There are two kinds of targets: Web and Powerscript. Both the Workspace and target concepts are aimed at the Web development environment. Targets are considered an excellent way of allowing you to “build, manage and deploy HTML, images and other

FIGURE 1 Migration Assistant

FIGURE 2 Migration Output Window

There are a number of objects now obsolete in PB 8, and some have moved to other places in PB. The release notes list these changes, and there are warnings as usual for obsolete objects that come with the migration. PB 8 has a few new concepts, but for the most part, the transition to this new system was straight ahead for a complex client/server environment.

Noticeable Improvement

The large number of objects involved only extended the time taken up by the transition between versions. The application is up and running, and seems to have gained a certain amount of speed between versions.

So far, all is well with PB 8 and EMS. ✅

mike.deasy@williams.com

Author Bio

Michael Deasy, a staff systems analyst for the Williams Companies in Tulsa, Oklahoma, has been working with PowerBuilder since version 3 and is president of the Tulsa PowerBuilder Users Group. Mike holds an MBA from Southern Nazarene University.

www.PowerBuilderJournal.com
n this article I cover the syntax for the new PB 8 exception handling, how to write the exception handlers that will improve your code, as well as issues concerning legacy code that currently implements exception-handling strategies.

If you’ve used Java or C++ to develop software, you’re probably well acquainted with exception handling in those languages. If you’re a PowerBuilder programmer, you may be familiar with the way previous versions of PowerBuilder handled exceptions in the SystemError event of the application object.

One of the more common sources of PowerBuilder SystemErrors occurs in relation to DataWindows. For example, this code is written in the clicked event of a DataWindow control:

```
dw_1.getitemstring(row,2)
```

It appears harmless, but it’s really a SystemError waiting to happen. The error will occur when you click on the DataWindow control, but not on a row within the control. In this case the value for row will be 0 and the SystemError will occur (see Figure 1). In this case the error is caught in the SystemError event of the application object.

You may already have code in the SystemError event to display the error and close the application, which is the default behavior for such errors, or you may decide not to close the application. In any case this level of error handling is more general than the new exception handling in PowerBuilder 8. In PB 8 exception handling has been improved to give the programmer more control at the moment in which the error occurs. Let’s start with the syntax:

Try-Catch-Finally

TRY statementClauseT

FIGURE 1 Invalid DataWindow row/column SystemError
The curly braces indicate optional parts. (If you’re a Java developer, don’t confuse the curly braces as tokens indicating code blocks.) Considering the optional possibilities from the previous text, it would appear that you could have just “TRY END TRY”; however, you need at least a CATCH or a FINALLY – you can’t exclude both. The ellipsis indicates that you can have multiple CATCH statements. As an example, multiple CATCH statements would look like:

```
Try
  //Your code here
Catch (MyThrowable t1)
  //Your code here
Catch (Throwable t2)
  //Your code here
End Try
```

The order of catching exceptions, from a more specific to a more general exception, is important. In the previous code, if the order were reversed and Throwable came before MyThrowable, PowerBuilder would give the error:

```
Error C0204: Catch statement unreachable. Ancestor (or same type) is caught in previous statement.
```

This is because MyThrowable is a Throwable and would never be reached, as Throwable would catch all exceptions intended for itself and its descendants.

**Getting Control**

Let’s continue looking at the clicked event of the DataWindow control as it provides a simple case study for our examination of the TRY-CATCH block. (I beg the reader to suspend comments about the quality of the code. Certainly it’s not necessarily the way such code should be written, but this simply allows us to examine the try-catch functionality.)

In Listing 1 we see that the object “Throwable e” has a text attribute that will contain the error message when the exception occurs. Executing this code won’t buy us a lot more than the error event in the application object. However, we do have more control over what happens next (see Figure 2). For example, we can correct the situation with the help of the user who clicked the control (see Listing 2). The exception gets invoked and we ask the user if he or she meant row 1. If the user responds in the affirmative, we can get the data from row 1 and move on as if nothing happened.

But wait, something evil is lurking in this “improved code”! Line 9 has:

```
ls_data = dw_1.getitemstring(1, "col1")
```

**Nested Exception Handling**

This brings us to the next topic since Listing 2, line 9 can cause a SystemError as well. Listing 3 provides an example for catching this “inner” exception. We’re beyond the simple case of the user clicking the wrong row. The name of the column is incorrect or has changed (another common mistake). In this case there’s nothing the user can do. Although some clever code could be written to attempt to determine the correct column name, it’s very problematic and not very feasible.

**Custom Exceptions**

The object type Throwable is the root data type for all user-defined exception and SystemError types. Two other system object types, RuntimeException and Exception, extend Throwable. Error types that derive from RuntimeException are typically used by the system to indicate runtime errors. RuntimeExceptions can be caught in a TRY-CATCH-FINALLY error script, but it’s not necessary to declare where such an error condition might occur (PowerBuilder does that for you). It’s also not required to catch these types of errors. Objects derived from exception are known as checked exceptions, user-defined exceptions that must be caught in a TRY-CATCH error handler when thrown or declared in the prototype of a method when thrown outside of a TRY-CATCH error handler.

There will be times when you need something more than the base class Throwable. For your own exception class, simply create a new standard class object in PowerBuilder that extends Exception, for example (see Figure 3).

I’ve called my new Exception class n_customer_exception to indicate that it will handle exceptions related to customers. However, I can’t use this new Exception object in place of my “Throwable” declarations because the system isn’t throwing an n_customer_exception.

To get around this problem and to move the code to a more maintainable place, I’ve created a nonvisual object (NVO) called n_customers to contain the code logic. The NVO has a method called of_getNames() that’s declared as throwing n_customer_exception (see Figure 4).

We see that the code in of_getNames() creates a n_customer_exception object and uses that to throw the exception (see Listing 4); the caller, which is in the newly modified clicked event of the DataWindow control, has the responsibility to catch it (see Listing 5).

**IS THAT YOUR FINAL ANSWER?**

Sorry, couldn’t resist. The last part of the TRY-CATCH-FINALLY block, appropriately named, is FINALLY. The code in the FINALLY clause is guaranteed to execute if any portion of the TRY block is executed, regardless of how the code in the TRY block completes. The FINALLY block is normally used to clean up after a TRY or
CATCH clause, for example, to close open file handles, if any.

If you’re a longtime PB developer like myself, you may often be tempted to destroy your created objects. But PowerBuilder will clean up any unreferenced and orphaned objects through its Garbage Collection mechanism. In Listing 5 the FINALLY clause attempts to destroy the thrown n_customer_exception object (the code was commented out to protect the unwary). Destroying a newly created “thrown” object is a huge no-no. This will likely cause the runtime to GPF because PowerBuilder isn't finished using the object.

**Legacy Code in the Application Object/Error Event**

Most likely you’re already handling your exceptions in a global manner by using a custom object in the Error event of the Application object that writes audit trails of the error out to a file and may HALT the application depending on the severity of the error. I recommend that you leave it alone based on what Sybase says:

“You should still have a SystemError event coded in your Application object to handle any uncaught exceptions. The SystemError event essentially becomes a global exception handler for a PowerBuilder Application”. Based on this advice, then, if you haven’t coded the SystemError event, you should, and use the new TRY-CATCH-FINALLY code when you want to “override” the SystemError event or to THROW your own exceptions, as we’ve seen.

**Conclusion**

With PowerBuilder 8, PB takes another step closer to creating better and more robust applications by using the TRY-CATCH-FINALLY keywords as well as adhering more closely to the “best practices” of object-oriented programming languages such as C++ and Java. PowerBuilder 8 is one of the most robust versions of PowerBuilder that I’ve seen. Its “dot-zero” release indicates that PowerBuilder will continue being a superior rapid application development tool for a long time to come.

**Listing 1: Simple exception handling**

```powerbuilder
string ls_data
TRY
  ls_data = dw_l.getitemstring(row,2)
CATCH (throwable e )
  messagebox(dw_l.dataobject, e.text)
END TRY
```

**Listing 2: The user can be proactive about the error**

```powerbuilder
string ls_data //line 1
int li_ret //line 2
TRY //line 3
  ls_data = dw_l.getitemstring(row,2)//line 4
CATCH (throwable e ) //line 5
  li_ret = messagebox("Mouse Clicked", "Sorry, not sure which row you meant." + "-r-n" + "Did you mean the first row?", Question!, YesNo!)
if li_ret = 1 then
  //line 8
  ls_data = dw_l.getitemstring(1,icol1) //line 9
else //line 10
  return //line 11
end if //line 12
END TRY //line 13
```

**Listing 3: Nested Try-catch**

```powerbuilder
string ls_data
int li_ret
TRY
  ls_data = dw_l.getitemstring(row,2)
CATCH (throwable e )
  li_ret = messagebox("Mouse Clicked", "Sorry, not sure which row you meant." + "-r-n" + "Did you mean the first row?", Question!, YesNo!)
if li_ret = 1 then
  try
    //line 17
    ls_data = dw_l.getitemstring(1,"coll")
  catch (throwable e2)
    messagebox("Mouse Clicked", "Sorry, there is a severe problem, cannot continue: -r-n*e2.text +i-r-n+ & iPlease contact support!")
  end try
else
  return
end if
END TRY
```

**Listing 4: The code moved into n_customers.of_getNames()**

```powerbuilder
string ls_data
int li_ret
n_customer_exception n_cust_ex
TRY
  ls_data = a_dw.getitemstring(al_row, 2)
CATCH (throwable e )
  li_ret = messagebox("Mouse Clicked", "Sorry, not sure which row you meant." + "-r-n" + "Did you mean the first row?", Question!, YesNo!)
if li_ret = 1 then
  try
    ls_data = a_dw.getitemstring(1,"coll")
  catch (throwable e2)
    n_cust_ex = create n_customer_exception
    n_cust_ex.text = "Sorry, there is a severe problem, cannot continue: -r-n*e2.text
    throw n_cust_ex
  end try
else
  return
end if
Finally
//if ( isValid(n_cust_ex)) then
//destroy n_cust_ex   //donit do this ñ PB will crash hard!
//end if
END TRY
```

**Listing 5: The new code in the clicked event of the DataWindow control**

```powerbuilder
n_customers nvo_custs
nvo_custs = create n_customers
try
  nvo_custs.of_getNames(dw_1, row)
catch (n_customer_exception e )
    messagebox("Error", e.text)
END TRY
```
The Depth and Breadth of Education to Advance Your Professional Skills

The JDJEdge 2001 Conference & Expo provides your best opportunity to understand how Java technologies can solve enterprise challenges.

Four information-packed JDJEdge Tracks featuring leaders in Java Technologies

Track 1  J2ME – Micro Edition & Wireless
Cutting-edge sessions for software engineers and hardware specialists working on wireless solutions.

Track 2  J2SE – Standard Edition
General Java programming for corporate programmers developing full Java applications, including several introductory sessions.

Track 3  J2EE – Enterprise Edition
Advanced sessions for software architects, Web programmers, corporate developers and consultants developing server-based applications.

Track 4  Working with I-Technology
Technical and management sessions for business analysts, corporate systems managers, architects, project managers and CIOs.

Who Should Attend
- Developers, Programmers, Engineers
- I-Technology Professionals
- Senior Business Management
- Senior IT/IS Management
- Analysts, Consultants

The only event backed by SYS-CON Media and Java Developer’s Journal

JDJEDGE 2001 Features
- Unmatched Keynoters and Faculty
- Over 150 Intensive Sessions and Tutorials
- The Largest Java Expo on the East Coast Participation by Invitation Only
- BEA WebLogic FastTrack to Certification
- IBM WebSphere FastTrack to Certification
- Macromedia ColdFusion FastTrack to Certification
- Sun Java FastTrack to Certification

www.sys-con.com/JDJEdge/
PowerBuilder 8

8/12 Sybase announced the general availability of PowerBuilder 8 in conjunction with the celebration of the tenth anniversary of PowerBuilder [http://my.sybase.com/detail/1,3693,1012912,00.html]. The features in the new version are discussed in a number of articles in this same issue of PowerBuilder Journal.

Usability Engineering Program

5/24 Sybase announced they are looking for test subjects for their usability engineering program, whom they will pay to participate. [http://my.sybase.com/detail/1,3693,1012826,00.html]

EAServer Svenska PA System of Sweden

5/23 Sybase announced an OEM partnership with Svenska PA System AB of Sweden [www.svenska-pa.se/], a Swedish provider of specialized human resource management and payroll solutions, under which they will embed Sybase EAServer into their human resource management software.

PowerDesigner ‘Neptune’ Debuts

8/04 Sybase demonstrated the next version of PowerDesigner – code named “Neptune” – at the 2001 JavaOne Developer Conference in San Francisco. Sybase plans to launch the Neptune beta program within three months, and begin shipping the product in the fourth quarter of 2001.

Meta Integration

8/11 Sybase announced that joint testing conducted with Meta Integration Technology, Inc. ([www.metaintegration.net]), has confirmed Meta Integration’s Model Bridge software will enable customers to transfer analysis and design models from virtually any vendor's design-tool software to Sybase PowerDesigner. Customers can then transform their existing metadata into PowerDesigner data models.

Mobile Automation

6/11 Mobile Automation Inc. ([www.mobileautomation.com]) announced that Anywhere Solutions agreed to license its MA2000 mobile device management technology. Mobile Automation’s MA2000 product is designed to facilitate the management of enterprise applications on a variety of devices, including laptops, PalmOS, and PocketPC handhelds.

Financial Fusion Server GlobalFIX Limited Edition

5/30 Financial Fusion, Inc. ([www.financialfusion.com]) – a subsidiary of Sybase – announced GlobalFIX Limited Edition (GlobalFIX LE), a functionality-light version of the market-leading Financial Fusion Server with GlobalFIX. Financial Fusion GlobalFIX LE is an electronic trading infrastructure solution that serves the needs of smaller firms and emerging electronic trading marketplaces, and is fully upgrade-compatible with the enterprise-class performance and high-availability features of Financial Fusion Server with GlobalFIX.

Industry Warehouse Studio for Media

8/04 Sybase ([www.sybase.com/iwsmedia]) announced the availability of Industry Warehouse Studio (IW/S) for Media, which allows newspaper, magazine, and cable TV companies to implement a complete customer relationship management (CRM) solution.

Events


DCI e-Business Portals Conference Sybase is a platinum sponsor July 30-31, 2001 - Boston, MA [www.dci.com/brochure/porphos]

Sybase TechWave 2001 Pre-Conference Courses August 11-12, 2001 San Diego Marriott Hotel and Marina [www.sybase.com/detail/1,3693,1012154,00.html]

The Data Warehousing Institute World Conference Sybase’s Business Intelligence Division is a sponsor November 4-9, 2001 - Orlando, FL November 26-30, 2001 - London, UK [www.dw-institute.com/confmain.htm]
Plan to Attend the 4-Day Conference
SHAPE YOUR WIRELESS STRATEGY
SAVE THE DATES!
WirelessEdge will provide the depth and breadth of education and product resources to allow companies to shape and implement their wireless strategy. Developers, i-technology professionals and IT/IS management will eagerly attend the WirelessEdge program to plan short and long-term wireless strategies to build competitive advantage. You will network with peers and explore cross-industry applications.

Plan to Exhibit
PROVIDE THE RESOURCES TO IMPLEMENT WIRELESS STRATEGY
RESERVE YOUR EXHIBIT SPACE NOW!
The Expo will provide access to the most qualified audience of potential buyers and specifiers to over 200 exhibitors.

The Conference will motivate and educate. The Expo is where attendees will want to turn ideas into reality. Be ready to offer solutions.

Plan to Sponsor
THE LARGEST WEST COAST WIRELESS CONFERENCE & EXPO IN 2002
MANY SPONSORSHIPS ARE EXCLUSIVE
Rise above the noise. Establish your company as a market leader. Deliver your message with the marketing support of SYS-CON Events and SYS-CON Media.

Conference Tracks

Track One: Development
- WAP
- i-Mode
- Bluetooth / 802.11
- Short Messaging
- Interactive Gaming
- GPS / Location-Based
- Wireless Java
- XML & Wireless Technologies

Track Two: Connectivity
- Smart Cards
- Wireless LANs incl. Bluetooth
- UMTS/3G Networks
- Satellite Broadband

Track Three: Wireless Apps
- Education
- Health Care
- Entertainment
- Transport
- Financial Services
- Supply Chain Management

Track Four: Hardware
- Cell Phones/WorldPhones
- PDAs
- Headphones/Keyboards/Peripherals
- Transmitters/Base Stations
- Tablets

Track Five: Business Futures
- Wireless in Vertical Industries
- The WWWW Unwired Management
- From 3W to 4W: Issues and Trends
- "Always-On" Management
- Exploiting the Bandwidth Edge
- Unplugged Valueware
- Wireless Sales & Marketing

Call 201 802-3069 to reserve your space today!