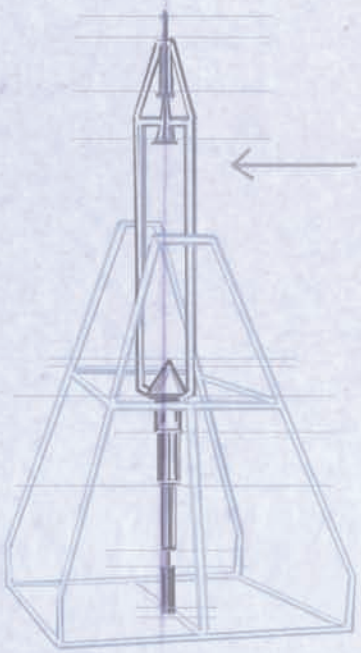
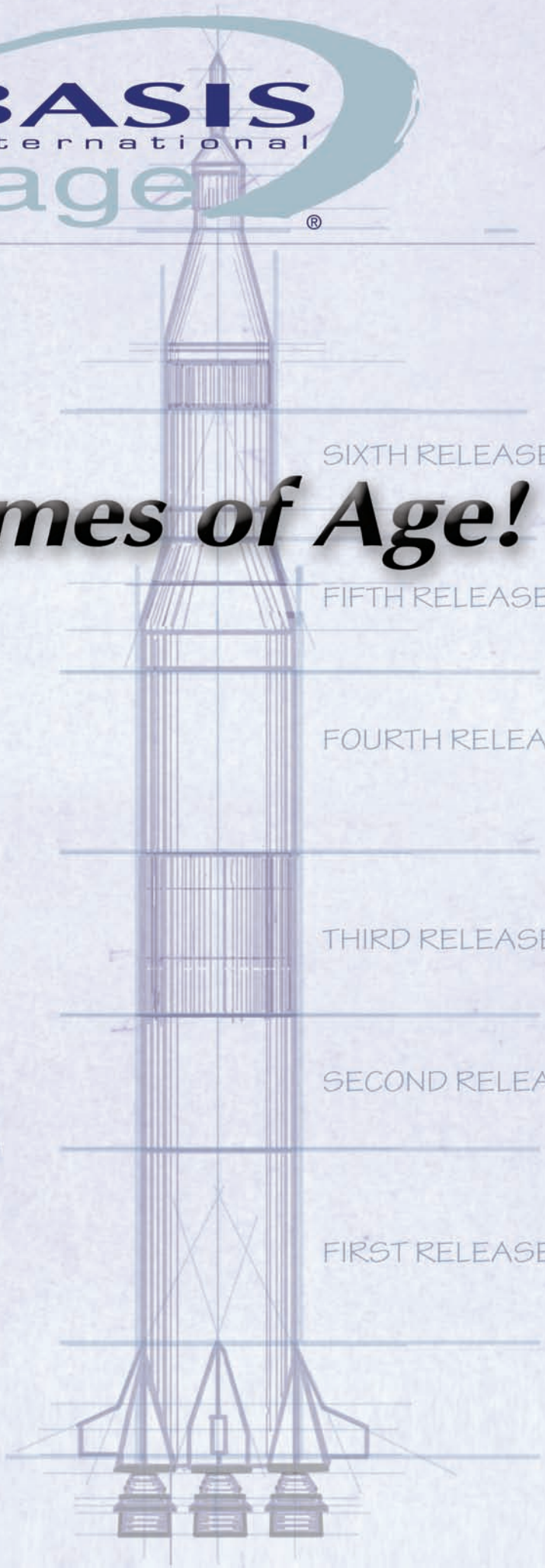
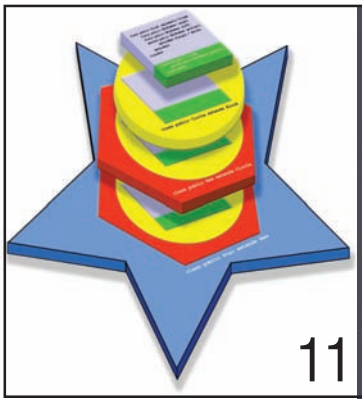
BASIS International
Advantage®

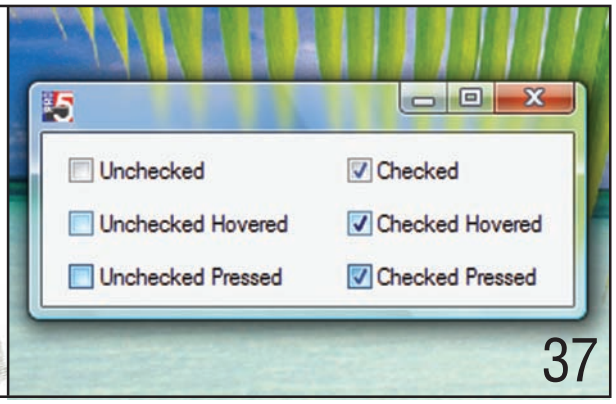
Number 1 • Volume 10 • 2006

BBj® Comes of Age!









Partnership

Language/Interpreter

DBMS

Development Tools

System Administration

COLUMNS

NEWS

e-article published exclusively at www.basis.com

For more information
Read about the new e-articles in *Gain the Advantage – Use the Advantage Resource* on page 18 of this magazine

BASIS Enters a Higher Orbit

The excitement at BASIS generated by the latest release of BBx® is truly infectious. Customers ranging from self-programming end users to large vertical market software houses are enthusiastic about the major new features available with BBj® 6.0. This release incorporates improvements and additional functions to each of the major areas of the BASIS product components; language/interpreter, DBMS, IDE, and system administration. This hardcopy issue and the extended online version overflows with articles that introduce and explain these new capabilities.

The language received a major improvement with the addition of full object-oriented functionality. This functionality gives our community the ability to attract and retain new human resources for enhancement and support of existing application solutions by providing syntax familiar to modern programmers. Read the article by one of our most respected Java specialists where he "outs" himself as a BBj fan by expounding on why he would choose BBj over Java for business application development.

BASIS embellished our DBMS, long the solid foundation of tried and trusted business solutions, with new advanced table types including a high performance exclusive SQL table type, and eXtended it with the industry standard database management features, triggers and stored procedures. Additionally, we took triggers and stored procedures to new heights by infusing the full power and functionality of the BBx language to these DBMS capabilities.

The new AppBuilder component of the IDE integrates with the existing FormBuilder module to deliver a powerful rapid GUI application development tool. This GUIBuilder replacement is able to read and write the existing .gbf file type while delivering a far more intuitive interface with extended functionality.

Take time to read the tongue-in-cheek introduction to the system administration and language article on resolving record-locking 'whodunit' puzzles with new enterprise management objects.

This issue also highlights BASIS' commitment and support of 64-bit operating systems for PRO/5® 6.0 and BBj 6.0 ports while bringing the Windows XP and Vista look and feel to Visual PRO/5® 6.0. These two features can bring enhanced performance and a new look to your applications with little or no effort. Moving your application from CUI to Classic Windows GUI took a lot of programming effort, but moving your application from Classic to XP or Vista is simply an upgrade!

The latest version of the BASIS product set is truly a major milestone in BASIS' vision of the delivery of modern language, database, and development tools. There is no time like the present to launch your application development efforts and blast into orbit with BASIS and the 6.x product line.

Nico Spence
Chief Marketing Officer

Dr. Kevin King
Chief Information Officer

The *BASIS International Advantage* magazine is published and distributed by BASIS International Ltd.

Editor in Chief *Nico Spence*
nspence@basis.com

Editor *Susan Darling*
sdarling@basis.com

Technical Editors *Dr. Kevin King, Nick Decker*
kking@basis.com, ndecker@basis.com

Copy Editor *Peggy Lewis*
plewis@basis.com

Art Director, Graphics *Patricia Catlett*
pcatlett@basis.com

Webmaster *Greg Smith*
gsmith@basis.com

Printing/Distribution Services
Albuquerque Printing Company

BASIS Corporate Portraits
Dale Frederick Photography

Advantage Subscriptions
www.basis.com/advantage/subs.html

BASIS International Ltd.
5901 Jefferson Street NE
Albuquerque, NM 87109-3432

Phone +1.505.345.5232
US Sales 1.800.423.1394
International +1.505.338.4188
www.basis.com

General Information
info@basis.com

Using Triggers to Maintain Database Integrity

By Jeff Ash

Most applications that manipulate and query data use some type of validation to assure data integrity. Currently, whether using WRITE RECORD operations or SQL INSERT and UPDATE statements in an interpreter session, all BBj® programs perform this validation in the application before writing the values to disk. However, what happens when an external ODBC or JDBC application such as a Web application performs write operations to that same data? The validation does not occur unless the ODBC or JDBC application duplicates that same validation routine. With BBj 6.0, developers can move the validation code to the table or file level, so that it occurs regardless of the method used to provide access to the table (e.g. WRITE RECORD, REMOVE, READ RECORD, or using SQL).

What is a Trigger?

A trigger consists of a block of BBj code that executes when a particular type of operation occurs on a particular data table. In other database management systems, a trigger belongs to a particular table in the database defined in the data dictionary. Since many BBj applications still use direct table or file access calls such as WRITE RECORD and READ RECORD that do not require a Data Dictionary definition, triggers belong to individual physical files. This means that a WRITE RECORD operation on a file and an SQL UPDATE or INSERT statement on a table defined in the Data Dictionary both cause write triggers to fire.

Real-World Example

A real-world trigger example should help illustrate trigger concepts and spark some ideas for other uses in your own applications. This example uses the Chile Company database included in the BBj installation. Suppose the Chile Company application creates a new order and then writes the order header information to the **`ord_hed`** file. Before adding the order, we want to check the validity of the customer number. If the customer number is not valid, the write operation should fail. This check should take place regardless of whether the operation originates from the BBj application or through a Web application using ODBC or JDBC.

The following steps walk through the process of adding a trigger to the **`ord_hed`** data file as shown in **Figure 1**:

1. Log in to the BBj Enterprise Manager.
2. Right click on the **Triggers** folder and select "Mount Directory of Files."
3. Navigate to the **`[bbj_install_directory]/demos/chiledd/data`** directory and click the [Select] button.
4. With the **Triggers** folder expanded, right click on the newly mounted directory name and select "Add Trigger."
5. Locate the **`ORD_HED`** file and select it from the file chooser.
6. With the newly mounted folder expanded, select the **`ORD_HED`** node in the Enterprise Manager tree.
7. Ensure the "Use The Triggers On This File" checkbox is checked.
8. Select "Before Write" in the Trigger Types list.
9. Ensure the "Enabled" checkbox is checked.
10. Click the [Save Changes] button.
11. Use the BASIS IDE or another text editor to save the code sample in **Figure 2** (available for download) listed below in a file named **`ORD_HED.trigger.bw.src`** and located in the same directory as the **`ORD_HED`** data file:

continued...

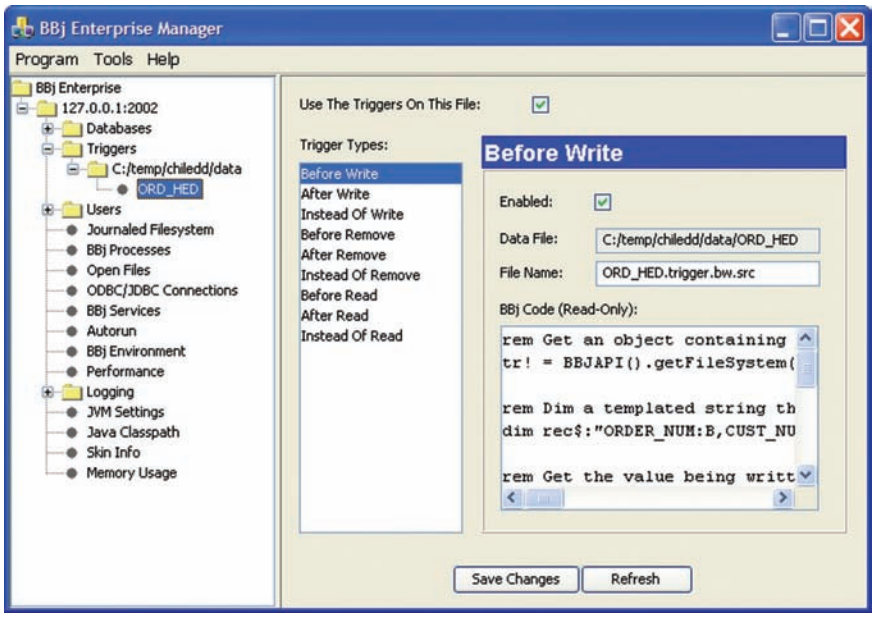


Figure 1. Trigger dialog box

Jeff Ash
Software Engineer

```
rem Get an object containing the trigger information.
tr! = BBJAPI().getFileSystem().getTriggerData()

rem Dim a templated string that matches the layout for ORD_HED
r$ = "ORDER_NUM:B,CUST_NUM:C(6),ORDER_DATE:N(7),SHIP_DATE:N(7),SHIP_ZONE:C(2),"
r$ = r$ + "SHIP_METHOD:C(5),COMMENT:C(30),SALESPERSON:C(3),NDSE_TOTAL:N(12),"
r$ = r$ + "TAX_TOTAL:N(12),FRGHT_TOTAL:N(12)"
dim rec$:r$

rem Get the value being written to disk
rec$ = tr!.getWriteBuffer()

rem See if the CUST_NUM is valid by ensuring it's in the customer data file
custNum$ = rec.cust_num$
chan = sqlunt
sqlopen(chan)"ChileCompany"
sqlprep(chan)"select count(cust_num) REC_COUNT from customer where cust_num = ?"
sqlexec(chan)custNum$
dim result$:sqltmpl(chan)
result$ = sqlfetch(chan)
sqlclose(chan)

rem If the CUST_NUM doesn't exist, throw an error
if result.rec_count = 0 then throw "Invalid customer number.",17
```

Figure 2. Code sample **ORD_HED.trigger.bw.src**

To test the new trigger, run the following UPDATE statement from a BBj program using the SQL verbs or from an ODBC or JDBC program such as Microsoft Query (**Figure 3**). Notice how the trigger causes the update to fail with our custom error message when using an invalid customer number.

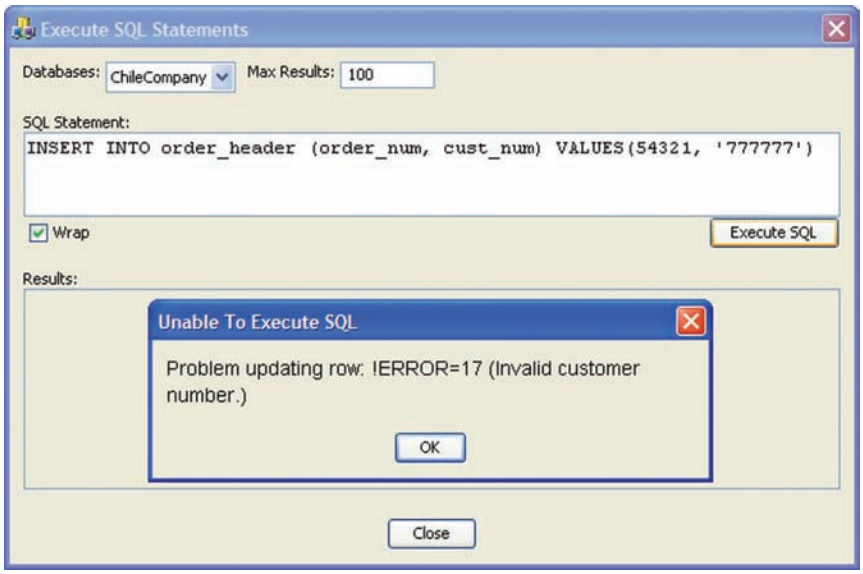


Figure 3. Custom error from trigger

Other Uses for Triggers

There are a number of other uses for triggers. Consider read and write triggers to encrypt/decrypt file data automatically. Another use might be a write trigger that logs information to a table that tracks who performed write operations on a file or a trigger that inserts data into a table in another database to keep the two databases in sync.

Summary

Most applications perform data validation inside the application itself. However, in BBj 6.0, developers can move data validation to a trigger attached to the data file. Using triggers makes this validation code available to everyone who accesses the data file, regardless of whether it is from a BBj application or a third party ODBC or JDBC application. When the code is at the file level, it provides a mechanism for much more consistent and reliable validation of data and therefore provides a solid mechanism for maintaining data integrity in your BASIS DBMS.

Download the sample program referenced in this article from
www.basis.com/advantage/mag_v10n1/triggers.zip

Using Stored Procedures to Add Business Logic to the Database

By Jeff Ash

Developers use a variety of methods and tools for building BBj® applications. However, the applications all consist of one or more programs that provide various functionality to the complete project. In some cases, it is beneficial to move some of the business logic into the database so that it is also accessible from third party ODBC or JDBC applications. BBj 6.0 provides this ability using stored procedures.

What is a Stored Procedure?

A stored procedure consists of a function or procedure written in BBj and embedded in the database. A new data dictionary file contains stored procedure information such as a definition of the parameters and features of the procedure, and points to a file containing the source code for the procedure. To access a stored procedure, a program executes an SQL CALL statement that specifies the name of the procedure and any necessary parameters. Using SQL makes it possible to call the procedure from BBj or any third party ODBC/JDBC application and get the same results.

Real-World Example

The best way to understand stored procedures and how they work in BBj is to walk through a simple real-world example. This example uses the Chile Company database included in the BBj installation. Suppose an application needs to acquire a list of customers who have ordered a particular item. Then suppose both a BBj application and a Web-based application need to access this information. One solution would be to provide all users with the appropriate SQL query so that the BBj and Web applications could use it. This would work fine until a change occurs in the query or the logic becomes more complex such that the query would not be able to provide the required result set. We could provide the changes to everyone so that they can update their code or we could use a stored procedure and centralize the logic.

The name of the stored procedure sample will be **LIST_CUSTOMERS** and will take a single argument – the item number for a product. Start by creating a new stored procedure in the Chile Company database:

1. Log in to the BBj Enterprise Manager.
2. Expand the **Databases** tree folder.
3. Locate the **Chile Company** database and expand its folder.
4. Locate the **Stored Procedures** folder and expand it.
5. Right click on the **Stored Procedures** folder and select **New Stored Procedure**.
6. Enter **LIST_CUSTOMERS** when prompted for the name of the procedure.
7. Click on the newly created **LIST_CUSTOMERS** node under the Stored Procedures folder.
8. In the right hand pane, enter **(DICTIONARY)list_customers.prc** for the Source Location field.
9. Place a check in the "Has Result Set" check box.
10. In the Parameter Specification list, click in the first cell and type the parameter name **ITEM_NUM**, set the SQL Type to **CHAR**, set the Direction to **IN**, and set Precision to **6**.
11. Click the [Save Changes] button.
12. Use the BASIS IDE or another text editor to save the following code sample in **Figure 1** (available for download) in your dictionary directory with the file name **list_customers.prc**:

continued...

```
rem Get an object containing the parameter values passed into the procedure.
sp! = BBJAPI().getFileSystem().getStoredProcedureData()

rem Get the item number passed into the procedure
itemNum$ = sp!.getParameter("ITEM_NUM")

rem Open an SQL channel to run the query on. We want to use the same database.
chan = sqlunt
sqlopen(chan, mode="PROCEDURE")sp!.getDatabaseName()

rem Run the query that will get the appropriate information
sqlprep(chan)"select c.cust_num, cvs(c.last_name,3) + ', ' + cvs(c.first_name,3)
:   cust_name from customer c, order_header o, order_line l
:   where l.item_num = ? and o.order_num = l.order_num and c.cust_num = o.cust_num"

sqlexec(chan)itemNum$

rem Set the returning result set to be the result of the query.
sp!.setResultSet(chan)
```

Figure 1. Code sample **list_customers.prc**

Jeff Ash
Software Engineer

To test the example, open Microsoft Query or another third party ODBC/JDBC application and enter the SQL statement shown in the dialog box in **Figure 2**.

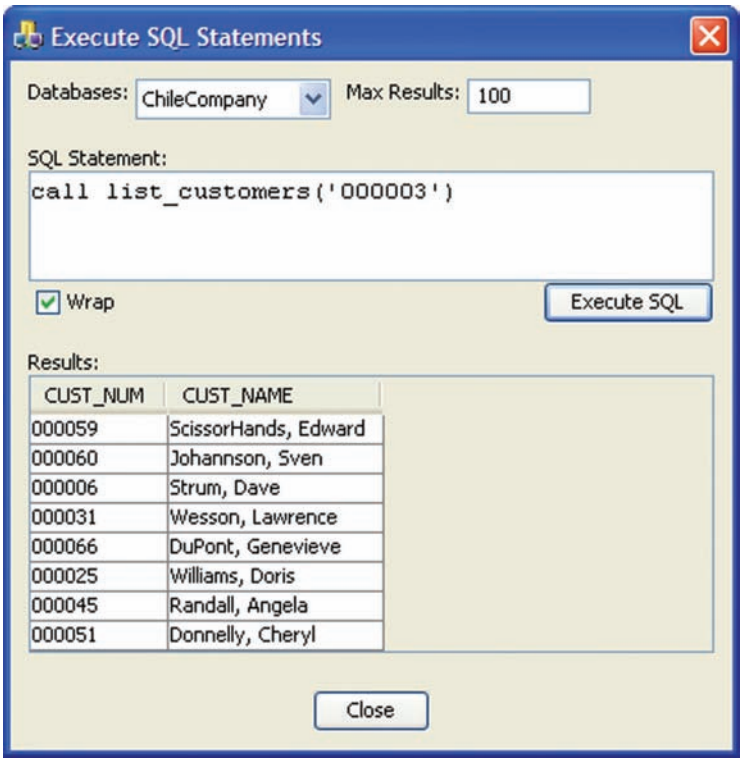


Figure 2. SQL statement

To call the procedure from a BBj application, execute **`sptest.bbj`** in **Figure 3** (also available for download):

```
rem Open up an SQL channel to the ChileCompany Database
rem and call the LIST_CUSTOMERS stored procedure

chan = sqlunt
SQLOPEN(chan) "ChileCompany"
SQLPREP(chan) "CALL LIST_CUSTOMERS('000010')"
SQLEXEC(chan)
DIM REC$:SQLTMPL(chan)

rem Print out the result set from the stored procedure
WHILE 1
    REC$=SQLFETCH(chan,end=*BREAK)
    PRINT REC$
WEND
```

Figure 3. Code sample **`sptest.bbj`**

Summary

Stored procedures offer a powerful new way for developers to add value and ease of maintenance to their BBj applications. Moving some of the common business and data access logic out of the BBj program into the database centralizes functionality in one place, making it accessible to the BBj program as well as any third party JDBC/ODBC application. Developers can extend the simple stored procedure example employed in this article to include complex BBj processing including CALL's to other BBj programs. All the powerful functions and features of the BBj language become available via an SQL CALL statement. A stored procedure's server-side processing delivers significant performance improvements over executing complex queries across the network as only the result set passes back across the network to the calling SQL statement. We encourage you to explore the myriad of new opportunities that you can take advantage of with this new BASIS DBMS feature.

Download the sample programs referenced in this article from
www.basis.com/advantage/mag_v10n1/sprocs.zip

A Primer for Using BBj Custom Objects

By David Wallwork

During the last several years, BASIS added a series of enhancements to BBj® that provide new functionality ranging from embedded Java to transaction-based commit and rollback; from Web Services that execute BBj applications to new development tools such as the IDE and AppBuilder; from new table or file types to data entry validation. Conspicuously absent from this list was the ability to write object-oriented code as BBj native code without using embedded Java.

BBj Custom Objects

BBj 6.0 introduces a number of new verbs that allow the developer to write classes and objects within a BBj program file, delivering fully object-oriented capability to Business BASIC. Classes defined in BBj are referred to as custom classes and instances of a custom class are referred to as custom objects.

Custom objects in BBj provide a full set of object-oriented features including:

- Protection levels (private, protected, public)
- Inheritance structures (extends)
- Interfaces structures (implements)
- Strongly-typed parameter lists and return values
- Multiple constructors
- Overloaded methods (accepting different parameter lists)
- Static variables
- Static methods
- Variable initialization
- Variable scope

A Custom Class

Understanding custom classes requires both a general knowledge of object-oriented programming and specific understanding of BBj syntax. Since there is a large body of information already published and readily available about object-oriented programming, we will not discuss the general concepts (see a few of these resources listed at the end of this article). What is important for us to demonstrate in this article is the syntax used in BBj to write an object-oriented program. The following two short examples demonstrate that syntax. A complete syntax definition may be found in the references at the end of this article.

The first example shown in **Figure 1** is a simple program that defines a custom class, instantiates an instance of that custom class (instantiates a custom object), and invokes a method on the instance. The output of this program prints **`hello world`** to the console.

```
declare Speaker speaker!
speaker! = new Speaker()
speaker!.speak()

rem definition of the CustomClass Speaker
class public Speaker
   method public void speak()
      print "hello world"
  methodend
classend
```

Figure 1. Sample program **`helloworld.src`** that defines a custom class

continued...

David Wallwork
Software Architect

The second example shown in **Figure 2** greets the user in Spanish, English, or French. Next, it prints information about the greeter and about the static value that holds the total number of guests greeted.

```
declare Greeter Pierre!
declare Greeter Joe!
declare Greeter Maria!
declare Greeter Host!

Pierre! = new FrenchGreeter("Pierre")
Joe! = new EnglishGreeter("Joe")
Maria! = new SpanishGreeter("Maria")
Host! = Joe!

while 1
    input "quel lingua?  0-espanol, 1-ingles, 2-frances   ", language

    switch language
       case 0
            Host! = Maria!
            break
       case 1
            Host! = Joe!
            break
       case 2
            Host! = Pierre!
            break
    swend

    Host!.greet()

    print "you have been greeted by: ", Host!.tellName()
    print "you are guest number: ", Greeter.getTotalGreetingCount()
    print "number of greetings by this Greeter: ", Host!.getGreetingCount()
    print
wend

class public Greeter

    field private BBjString Greeting$
    field private BBjString Name$
    field private BBjNumber count
    field private static BBjNumber totalCount

    method public Greeter(BBjString p_greeting$, BBjString   p_name$)
        #Greeting$ = p_greeting$
        #Name$ = p_name$
    methodend

    method public void greet()
         print
         print #Greeting$
         #count = #count + 1
         #totalCount = #totalCount + 1
    methodend

    method public static BBjNumber getTotalGreetingCount()
         methodret #totalCount
    methodend

    method public BBjString tellName()
         methodret #Name$
    methodend

    method public BBjNumber getGreetingCount()
        methodret #count
    methodend

classend
```

continued...

```
class public FrenchGreeter extends Greeter
     method public FrenchGreeter(BBjString p_name$)
        #super!("bon jour",  p_name$)
     methodend
classend

class public EnglishGreeter extends Greeter
     method public EnglishGreeter(BBjString p_name$)
        #super!("hello ",  p_name$)
     methodend
classend

class public SpanishGreeter extends Greeter
     method public SpanishGreeter(BBjString p_name$)
        #super!("buenos dias",  p_name$)
     methodend
classend
```

Figure 2. Code sample **greeter.src** that greets users in multiple languages

This small program demonstrates a number of object-oriented concepts and how they are realized in BBj. The reader will notice the use of polymorphic classes (the various child classes of Greeter), static values and static methods (totalGreetingCount), class extension, and protection levels illustrated in **Figure 3**.

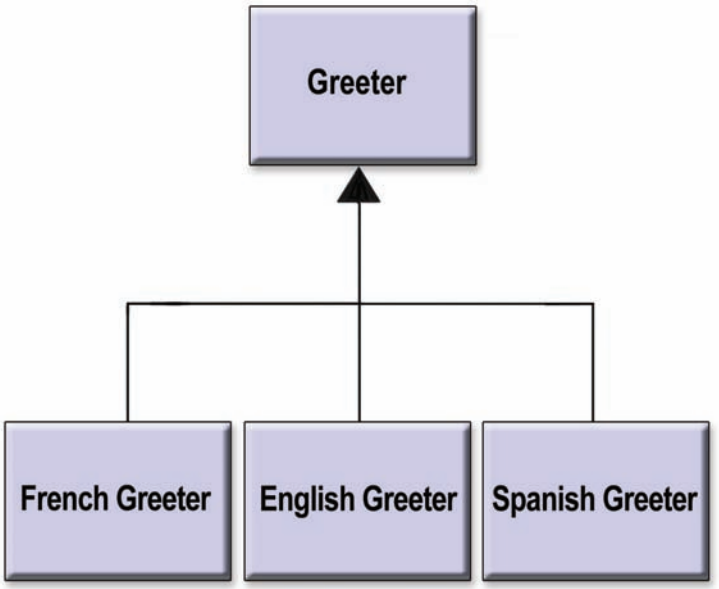


Figure 3. Using polymorphism in BBj

Summary

Though these examples assume an understanding of object-oriented programming, many readers may be unfamiliar with this concept. There is an abundance of well-written technical resources available, but BASIS recommends reviewing the sites listed below as a good starting point to learn the basic concepts.

More than any other of the changes in the recent past, the adding of object-oriented programming to BBx® technology has the greatest potential to impact the day-to-day development and the way developers build business applications. BBj custom objects can streamline the development process and promote code re-use, resulting in more robust applications getting to market faster than ever before. Truly, adding object-oriented programming takes BBx technology light-years ahead of traditional Business BASIC.

Download the sample programs referenced in this article from
www.basis.com/advantage/mag_v10n1/usingco.zip

For an overview of how object-oriented programming techniques can help you write modules that are more robust and reusable, see
java.sun.com/docs/books/tutorial/java/concepts

To learn more about custom objects and how to use them in BBj, refer to
www.basis.com/solutions/BBj_CustomObjects.pdf and
www.basis.com/solutions/BeginBBjObj.ppt

AppBuilder: The BASIS IDE Gets RAD GUI Development Integration

By Jon Bradley

The new BASIS IDE component, AppBuilder, has arrived. Debuting in BBj® 6.0 and coupled with the IDE's GUI screen design tool, FormBuilder, AppBuilder provides a much-anticipated cross-platform graphical application development tool. AppBuilder provides all the functionality BBx® developers know and love from GUIBuilder, but with a more intuitively designed interface delivering *what you see is what you get* (WYSIWYG) functionality. **Figure 1** shows a standard AppBuilder screen in all its glory.

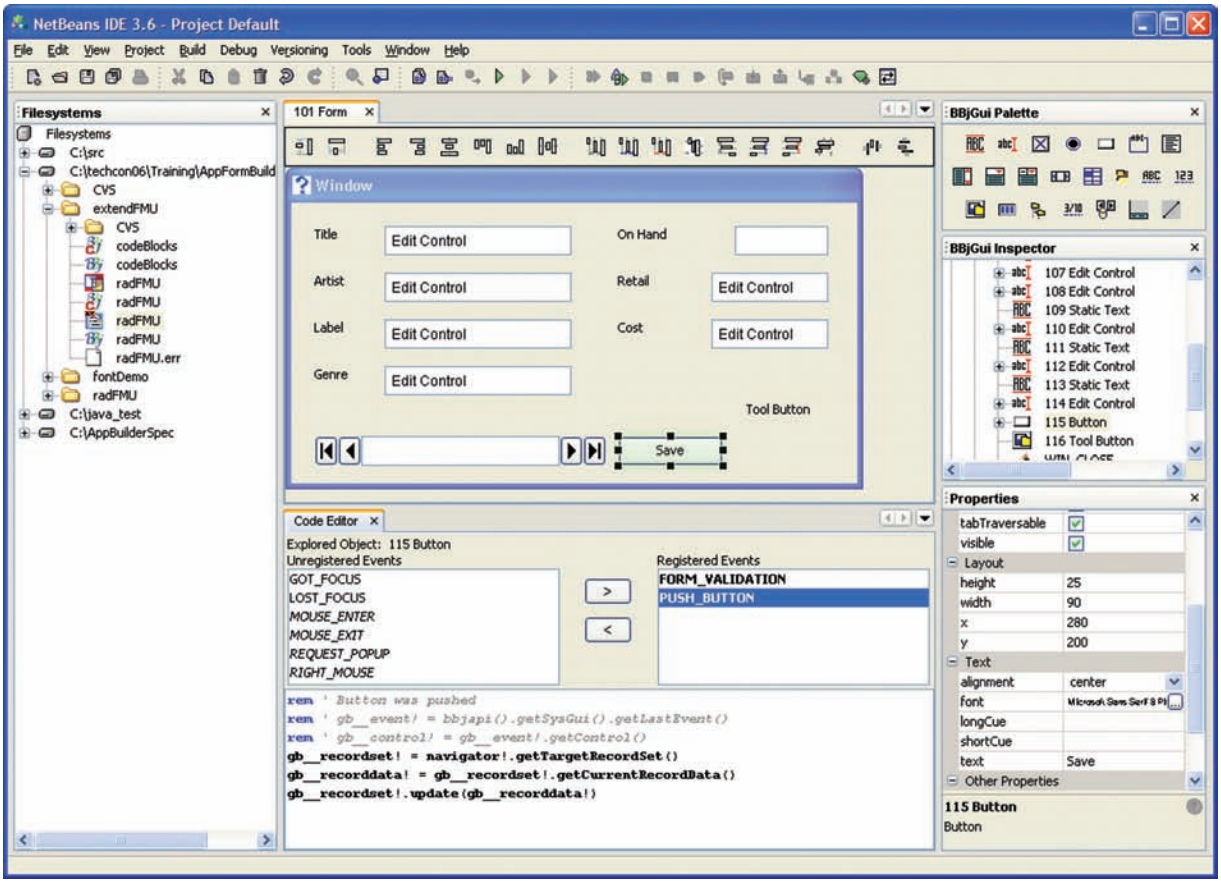


Figure 1. The AppBuilder desktop

AppBuilder makes it easier and faster to create graphical user interface (GUI) applications. A typical GUI program has many elements.

- Initialization
- Specification of what controls and windows the GUI shall display
- Specification of what user actions cause the execution of event handlers
- Event loop
- Error and escape handling
- Cleanup

AppBuilder makes development a snap by writing the general framework, often called the boilerplate code, which varies very little from application to application. By integrating tightly with FormBuilder, AppBuilder is a great tool for developing GUI resources and specifying what controls and windows display when the application runs. With FormBuilder and AppBuilder, the BASIS IDE allows the developer to edit the resource file while also specifying what the program does when the user interacts with those controls in a given way. By writing all the boilerplate code, AppBuilder reduces the difficulty and time required to write a BBj GUI program. In addition, AppBuilder only displays the developer-specified event handling code and drastically reduces the amount of code a developer wades through while working on the application.

continued...

Jon Bradley
Software Engineer

GUIBuilder Turbo Charged

AppBuilder is a functional replacement for GUIBuilder, BASIS' previous application development tool. The tight integration with FormBuilder and more intuitive interface are AppBuilder's primary advantages over GUIBuilder. The developer can now edit the resource and navigate through the event handling code blocks visually via the integration and coordination between the FormEditor, BBjGui Inspector, and Code Editor shown in **Figure 2**.

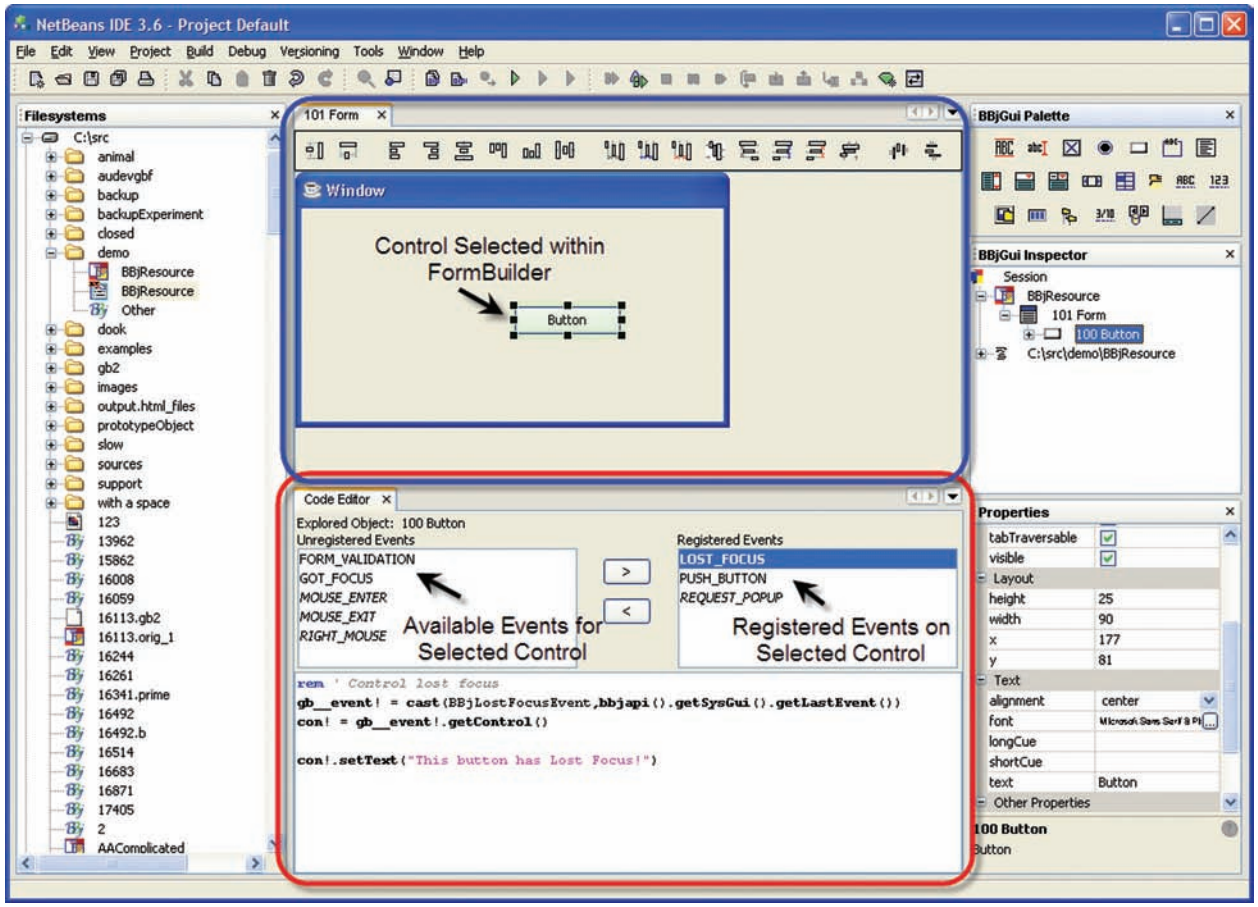


Figure 2. FormBuilder's Form Editor (**blue outline**) and AppBuilder's Code Editor (**red outline**)

Immediately after adding a control via the FormBuilder palette, AppBuilder's Code Editor displays the available and registered events. In addition to listing the events, the Code Editor provides a syntax-colored, code completion-enabled editor for editing a given event's code block.

The BBjGui Inspector component displays and navigates the registered events and controls within the application in a tree format (see **Figure 3**). Non-event specific code blocks, such as the Init block, are listed as well.

In addition to the updated graphical interface, AppBuilder provides several useful enhancements over GUIBuilder. Like many modern applications, pressing **[F1]** opens up context-sensitive help. Code completion further reduces reliance on external documentation by providing the developer with the method signatures for the BBjAPI objects on demand.

continued...

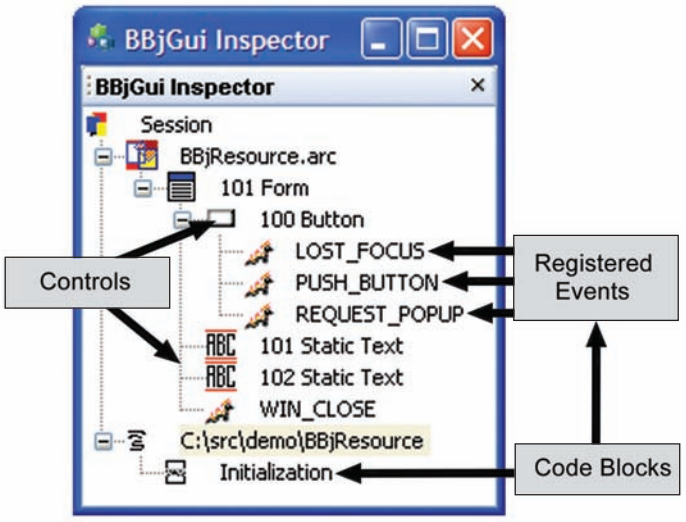


Figure 3. BBjGui Inspector

Automatically Inserting Commonly Used Code

Registering an event adds some default code for that event as specified by the Default Code Profile (see **Figure 4**).

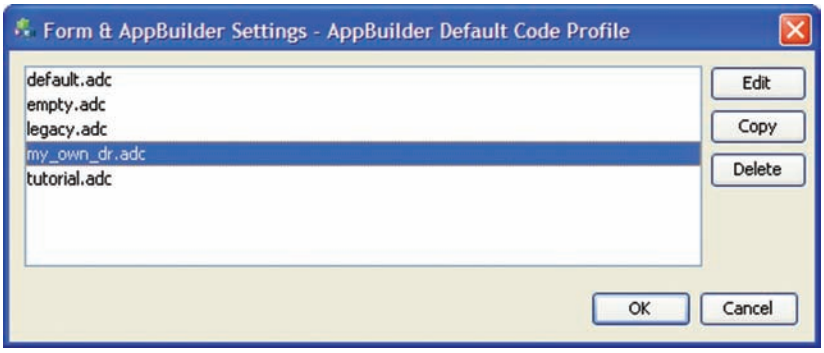


Figure 4. Default Code Profile dialog box

In addition to being able to configure what code AppBuilder inserts automatically when the developer registers for an event, the developer can configure the default code profiles to register automatically for specific events upon the addition of controls (see **Figure 5**).

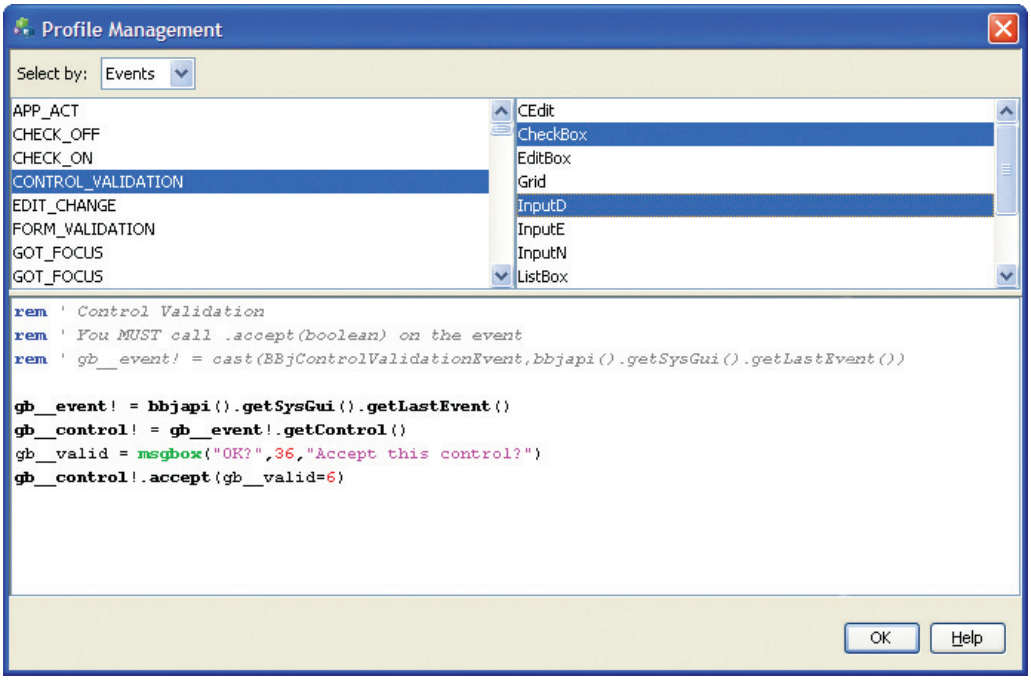


Figure 5. Profile Management dialog box

AppBuilder Additions to GUIBuilder Functionality

AppBuilder optionally utilizes a pre-processor to provide developers a chance to modify the AppBuilder file also known as a **`.gbf`** file. The pre-processor can easily perform string literal substitutions as shown in **Figure 6**.

Compatibility and a Look Towards the Future

Existing **`.gbf`** files created by GUIBuilder can be loaded directly in AppBuilder. AppBuilder is backwardly compatible with GUIBuilder and, therefore, Visual PRO/5® as it generates the same READ RECORD code as GUIBuilder. Since the language has progressed significantly since the introduction of GUIBuilder in 1998, BASIS plans to add a wizard for generating a GUI application from a defined record set. At a later date BASIS will update AppBuilder with the capability of generating modern code including event objects, callback event dispatching, and AppBuilder custom objects. This will provide developers a choice between backwards compatibility or the use of the powerful new language features.

continued...

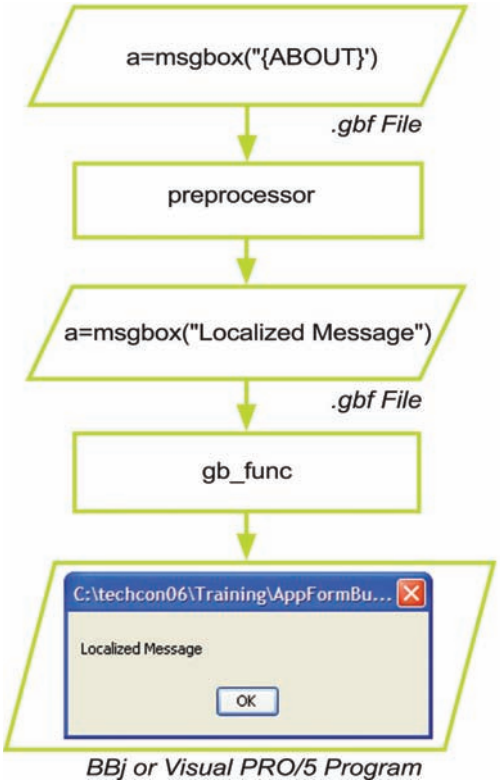


Figure 6. Sample string literal substitutions

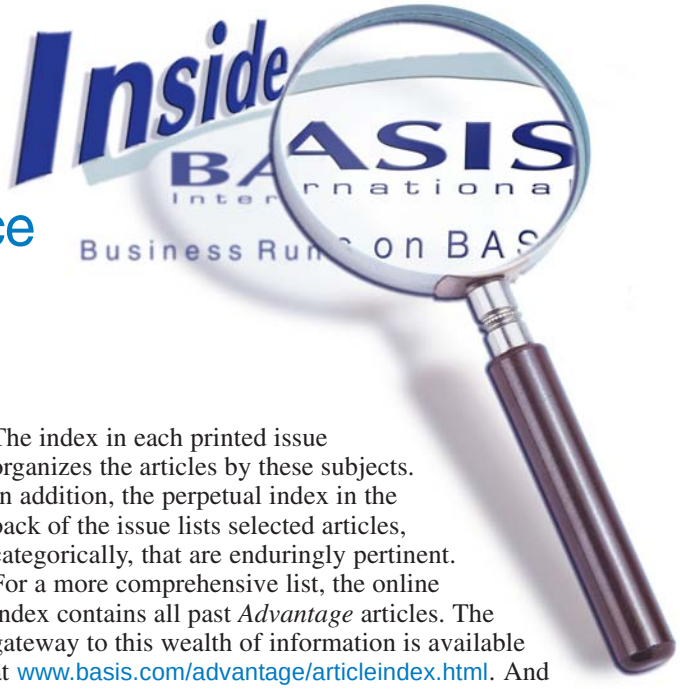
Summary

By providing a cross-platform application development tool integrated within the IDE, BASIS continues to improve developers' programming experience. By generating and hiding the boilerplate code, AppBuilder makes creating GUI applications faster and less repetitive. The integrated interface for defining GUI resources along with their behavior makes programming more intuitive. The addition of syntax coloring and code completion reduces programmer errors, and reduces reliance on documentation. Give AppBuilder a try, and see how it improves developer productivity, and how "easily" the development process unfolds.

For more information, read
FormBuilder: BASIS IDE's Better Cross-Platform Resource Builder at
www.basis.com/mag-v9n1/fb.html

Gain the Advantage – Use the *Advantage* Resource

By Greg Smith



The *BASIS International Advantage* magazine is the premier technical resource for the BBx® Generations. To continue delivering this valuable resource, BASIS made some format changes in recent issues and is introducing more enhancements in this issue.

The most noticeable addition was color-coded subject tabs in the margin of each article. These tabs help readers easily identify and focus on articles of interest. The subject tab labels appear below and include the four BASIS technology components.

Language/Interpreter

PRO/5®, Visual PRO/5®, and BBj® languages and their associated interpreters, enhancements and additions

DBMS

The BASIS Database Management System, table types, ODBC/JDBC, and data storage and retrieval

Development Tools

Application development and management tools including the BASIS IDE, AppBuilder, and FormBuilder

System Administration

Enterprise management and configuration, user access control, system resource management, installation, and managing BBx environments

Partnership

Successes of BASIS Partners, strategic initiatives to strengthen the partnership relationship, marketing, and technical support of BASIS partners

The index in each printed issue organizes the articles by these subjects. In addition, the perpetual index in the back of the issue lists selected articles, categorically, that are enduringly pertinent. For a more comprehensive list, the online index contains all past *Advantage* articles. The gateway to this wealth of information is available at www.basis.com/advantage/articleindex.html. And to locate a particular article easily, the "Advanced Search" selection on any BASIS Web site page now searches all online *Advantage* articles as well as the BASIS documentation, Knowledge Base, and entire Web site.

In an effort to provide readers with as much relevant information as possible without the space limitations of a printed magazine, this issue debuts articles exclusively available on the Web site. To identify the e-articles, the titles appear in the table of contents and index with a symbol in place of a page number.

With all these changes, readers can easily obtain information by category or a particular article of interest. The abundance of valuable information in the *BASIS International Advantage* is literally at your fingertips and will prepare and sustain you for victory in the front lines of application development, deployment and support. And THAT is the *BASIS Advantage* advantage.

For future reference, bookmark the complete index of the *BASIS Advantage* articles at www.basis.com/advantage/articleindex.html

Greg Smith
Webmaster

Confessions of a Language Polygamist

By Jon Bradley

Some programmers continually experiment with new languages and feel comfortable using multiple languages, even on the same day. I am not one of those people. I have always regarded myself as a *language monogamist*. In the same way that some people can chart their lives by the music albums they listened to in different periods of their lives, I can divide my life into stages based on the language in which I programmed. As a young rookie, I programmed in QBasic, then C. Somewhere in college I began to experiment, going from C to PERL. After a summer job tricked me into switching from PERL to Java, I thought Java was my "one and only - 'til death us do part."

My Confession

Well I was wrong. Over the course of several years as a BASIS Java engineer, BBj® began to move ahead and take its place as my second workhorse language. It all came as a bit of a surprise, but often BBj was the fastest way I could think of to implement some random application, be it a test harness, or a mortgage calculator. In many ways, BBj meets or beats Java by providing a sophisticated environment, an easy to use GUI, and some great development tools. This makes sense when you stop to think about it. BASIS wrote BBj in Java. The BBj project has many hundreds of thousands of lines of code; many of which specify improvements in Java functionality, or simplify tasks that are more difficult in Java.

At BASIS' TechCon, an attendee asked me "If I don't have any old PRO/5® code or data and if I'm starting from scratch, why would I use BBj instead of just using Java?" It then occurred to me that we do not highlight the improvements BBj makes upon Java. So, I took the mission upon myself to do just that – expose, in this article, the many ways that developing in BBj is BETTER than or AS good as developing in Java.

Simpler Code

In many cases, BBj code is simpler and shorter than Java. For instance, compare the standard **`Hello <name>`** program in BBj (**Figure 1**) with Java (**Figure 2**):

continued...

```
input "Please type your name: ",name$
print "Hello ",name$
```

Figure 1. **`HelloName.src`** code sample in BBj

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class HelloName
{
    public static void main(String args[]) throws IOException
    {
        String name;
        BufferedReader br = new BufferedReader(
                new InputStreamReader(System.in));
        System.out.print("Please type your name: ");
        name = br.readLine();
        System.out.println("Hello "+name);
    }
}
```

Figure 2. **`HelloName.java`** code sample in Java

Jon Bradley
Software Engineer

An even more revealing example is in a small GUI application illustrated in the next two figures. Compare the incredibly large amount of complex Java code in **Figure 3** with the smaller amount and far simpler BBj code in **Figure 4**. Two very different programs, one almost triple the size of the other, result in identical output.

continued...

```
import java.awt.Dimension;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.UIManager;

public class PushMe extends JFrame
{
    public static void main(String args[])
    {
        //kind of optional, but i wanted the two apps to look the same
        setupLookAndFeel();

        JFrame win = new JFrame();
        win.setTitle("Simple GUI App");
        win.setPreferredSize(new Dimension(200,200));
        win.setDefaultCloseOperation(win.EXIT_ON_CLOSE);
        win.getContentPane().setLayout(null);
        JButton button = new JButton("Push Me");
        button.addActionListener(new DialogPopper(button,win));
        button.setSize(new Dimension(120,30));
        button.setLocation(30,50);
        win.getContentPane().add(button);
        win.pack();
        win.setVisible(true);
    }

    /** Not necessary, but i wanted the programs to look the same,
     * and Java by default doesn't do the XP look and feel
     */
    private static void setupLookAndFeel()
    {
            try {
                UIManager.setLookAndFeel(
                        UIManager.getSystemLookAndFeelClassName() );
            } catch (Exception e) { }
    }
}

class DialogPopper implements ActionListener
{
    private final JButton m_button;
    private final JFrame m_frame;

    DialogPopper(JButton p_button, JFrame p_frame)
    {
        m_button = p_button;
        m_frame = p_frame;
    }

    public void actionPerformed(ActionEvent p_e)
    {
        int result = JOptionPane.showConfirmDialog(m_frame,
                "Disable the Button?", "Modal Dialog",
                JOptionPane.YES_NO_OPTION, JOptionPane.PLAIN_MESSAGE);
        if(result == JOptionPane.YES_OPTION)
        {
            m_button.setEnabled(false);
        }
    }
}
```

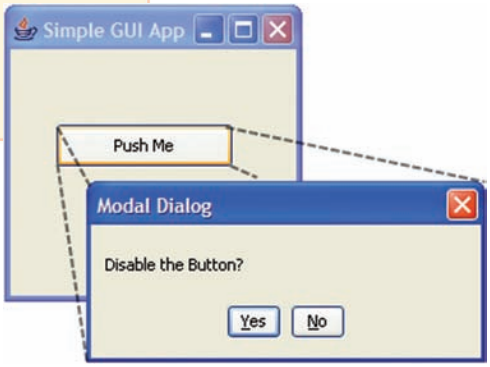


Figure 3. PushMe code written in Java to create the sample GUI output

```
SYSGUI=UNT
OPEN(SYSGUI)"XO"

declare BBjWindow win!
rem height is different because our height specifies client height, not frame height
win! =BBjAPI().getSysGui().addWindow(50,50,200,170,"Simple GUI App")
win!.setCallback(win!.ON_CLOSE,"APP_CLOSE")

declare BBjButton button!
button! = win!.addButton(101,30,50,120,30,"Push Me")
button!.setCallback(button!.ON_BUTTON_PUSH,"PUSHED")

PROCESS_EVENTS

PUSHED:
    if(MSGBOX("Disable the Button?",4,"Modal Dialog") = 6) button!.setEnabled(0)
return

APP_CLOSE:
    release
return
```

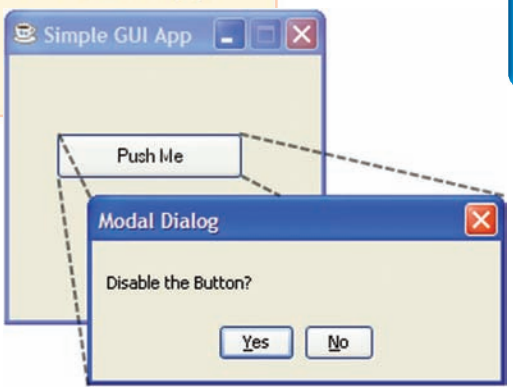


Figure 4. PushMe code written in BBj to create the same sample GUI output

As you can see from these small examples, the BBj code in **Figure 4** is more succinct. BBj provides a host of language functionalities that other languages, such as Java, typically leave to the developer. For instance, Databound GUI controls make the process of displaying and updating table/file information significantly less complicated in BBj.

Objects

An area in which BBj used to lag was its object oriented capabilities. Since 2006, BASIS has offered custom objects that provide the developer with a robust object system. In addition to custom objects, BBj has always been able to embed Java objects including the extensive core libraries, or custom Java classes.

Database Management System

Out of the box, Java does not provide any keyed table or file types or database capabilities. BBj, however, provides transaction tracking, extremely fast keyed look-ups, table/file locking, and full DBMS functionality. BBj provides SQL access via both the SELECT verb, and JDBC/ODBC access. BBj provides triggers and stored procedures to ease implementation of a host of data-centric issues. In addition, both trigger and stored procedure logic is coded with the BBj language.

Deployment Choices

Using 3-tier BASIS architecture, developers can deploy BBj in many different ways. Thin Client allows the data and processing to occur on a remote server, while the GUI or CUI presentation occurs on the client machine. Deploying through Web Start can update client machines automatically with the latest BBj.

Interactive Interpreter

In BBj, the developer can debug/edit a running application within our fully interactive interpreter. This greatly speeds development and maintenance of applications due to the ability to load or run arbitrary code at the command prompt. There simply is no **>READY** prompt in Java. For many of our developers this feature has been the essential enabler for delivering exceptional end-user support and troubleshooting capability.

Program Format

BBj can load or save both ASCII and tokenized programs. Furthermore, a developer can load a program as ASCII and save it as tokenized, or vice versa. In Java, not only is the source program always flat text, (with a required **.java** extension), but Java can only run a compiled **.class** file. BBj also provides SAVEP protection to protect your source code from prying eyes.

continued...

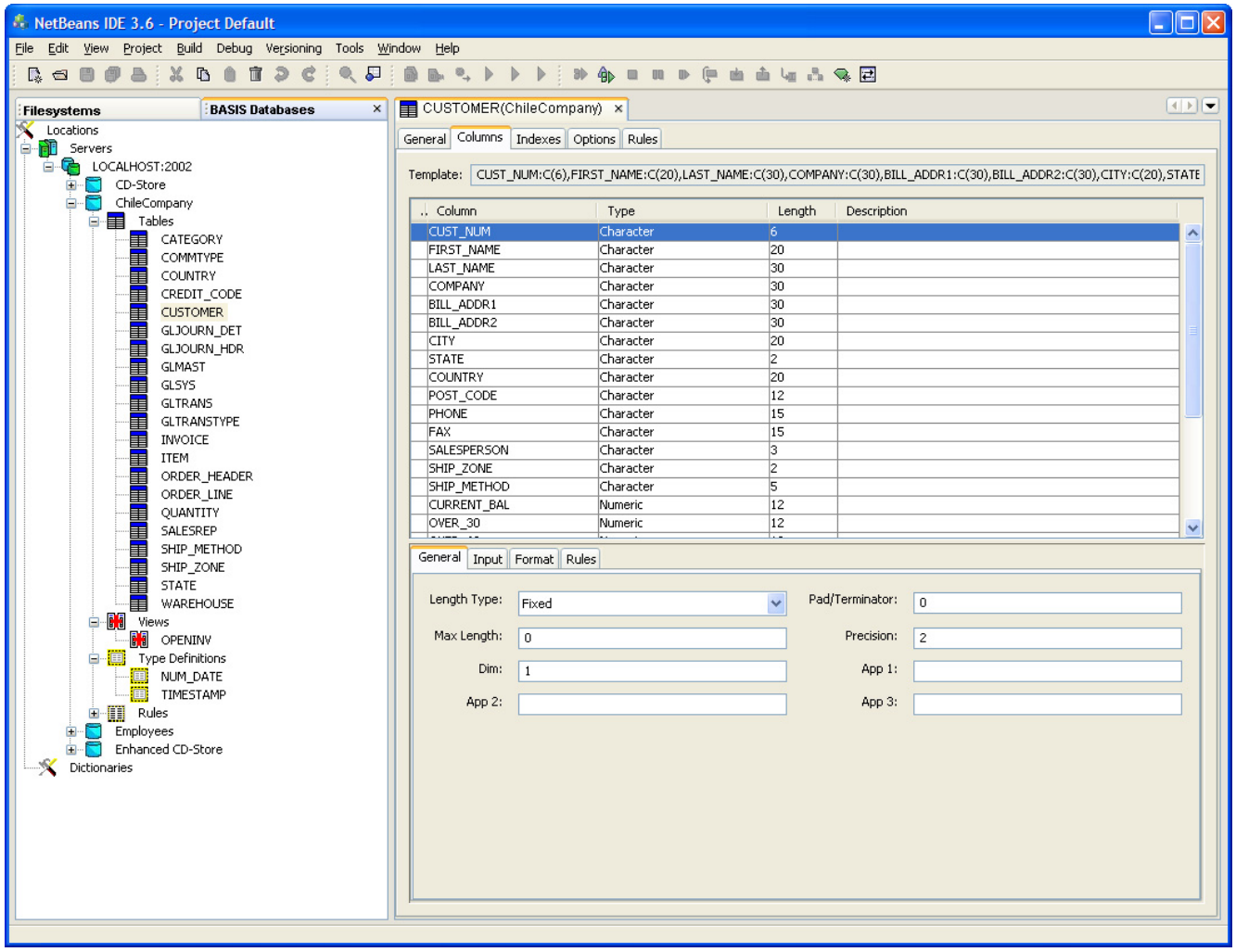


Figure 5. Configuring a database using the BASIS IDE

Developer Tools

BASIS facilitates development of applications with the BASIS IDE as shown in **Figure 5**. From within the IDE, you can edit and debug programs, configure Data Dictionary tables/files, create resource files to specify GUI forms, view data within BASIS-keyed tables/files, or quickly create full applications using AppBuilder.

Developer Support

Anyone active on the BBj developer list knows that BASIS provides very responsive support to developers. While Java does have a developers' forum, you can measure responsiveness to a bug report in 'months.' BASIS often provides turnaround from a support and sales standpoint the same day. Our engineering department frequently incorporates suggestions from the development community in upcoming releases.

Summary

So, it is no secret that I still program in Java, but there are so many situations where BBj's interactive interpreter, easy table/file access, and GUI system make it the easiest language to use. The BASIS IDE allows me to create GUI screens and full applications faster than I could in Java. With backward compatibility to legacy Visual PRO/5® code and data, BBj certainly has its own share of strengths when standing up against Java. I guess this does make me a language polygamist…enjoying the best of both worlds.

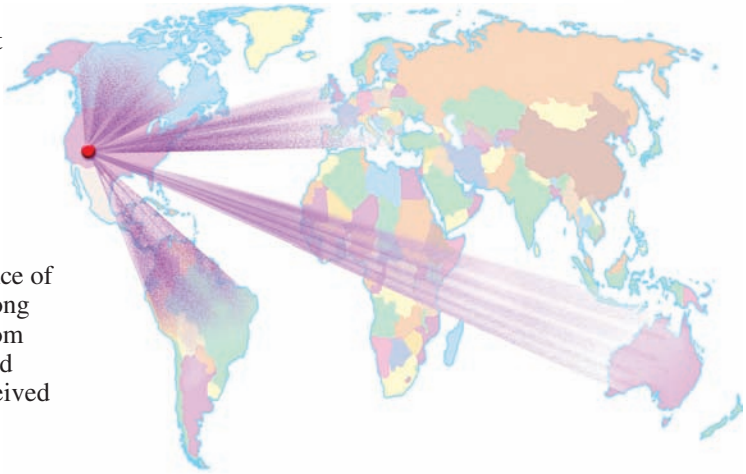
For additional information, read *Using Triggers to Maintain Database Integrity* on page 6 and *Using Stored Procedures to Add Business Logic to the Database* on page 8 in this issue.

For more information about deployment choices, read *Choices, Choices, Choices* at www.basis.com/advantage/mag-v9n2/choices.html

Webinar Success Impacts the Globe

By Laurence Guiney

Over 90 people registered for the recent Webinar sessions on the new DBMS Features and Benefits. Registrations came in so rapidly during the first 24-hours that BASIS added two more sessions, including an early evening session. BASIS chose this time to accommodate overflow registrations but more importantly, for the convenience of the BASIS resellers in Australia. Among the participants was representation from North and South America, Europe, and Australia. The sessions were well received and the feedback has been positive.



Strengthening the Down Under Connection

By Laurence Guiney

To build ties and strengthen business relationships with our Australian partners, BASIS now deals directly with their resellers "down-under." This relationship gives extra dividends to BASIS and participating BASIS customers when these partners shared valuable knowledge and sparked helpful dialog during recent Webinars. Several Aussie partners who are currently using PRO/5® technology are now investigating the possibilities of moving to the latest generation of BBx®, BBj®, to take advantage of the new technology in the language and BASIS Database Management System.

Signs of the Time - Digital Signatures

By Laurence Guiney

Staying on the cutting edge of technology, BASIS switched from mailing hard copy invoices to sending invoices electronically over two-years ago. Customers around the world appreciate this expeditious and efficient way to process their BASIS payables, not to mention the related cost savings.

While e-invoices were very well received, they presented a recent challenge in Germany. Unique to this country is Germany's requirement for an electronic signature on each invoice to verify its authenticity. This seemed like a worthwhile enhancement to all e-invoices so BASIS selected VeriSign, the leading secure sockets layer (SSL) certificate authority, as the provider for domestic invoices. German law required electronic signatures from an approved German provider so BASIS chose Dynevo, GmbH.

Today, all e-invoices now successfully contain a signature that can be validated for authenticity. Regardless of their location, BASIS customers around the world benefit from this paperless and cost-effective method of invoice delivery. Keeping up with the ever-changing technology and needs of the customers is a key objective for BASIS doing business in the US and abroad.

Laurence Guiney
Senior Account Manager

The BASIS Family in Europe Grows

By Susan Darling

Former owner and president of PHAROS Ltd., Stephan Wald, is the new Director of Sales and Technical Services of BASIS Software Germany (BSG) GmbH. Stephan will lead BASIS' European distribution and service business, extending and enhancing the existing reseller network and supporting strategic customers.


Stephan Wald

Stephan is no stranger to BASIS or BBx®. In 1996, Wald founded PHAROS Ltd., Saarbrücken, Germany, to develop and support customers in the Business BASIC environment. In the ensuing years, PHAROS achieved an excellent reputation among its broad customer base throughout Germany and bordering France. In 2005, BASIS selected PHAROS as their pre- and post-sales support partner in France.


Andreas Timm


Angela Laermann

While the BSG office in Wiesbaden will continue to provide the European distribution of the BASIS development tools, the remaining PHAROS Ltd. team is integrating into BASIS as a new business unit known as *BASIS Professional Services*. Joining BASIS along with Stephan, in this new entity, is Andreas Timm and Angela Laermann. Andreas, fluent in German, French, and English, earned an advanced degree in Computer Science. Angela, also fluent in German, French, English, as well as Polish, also holds an advanced degree in Computer Science.

Applying Custom Objects to Existing Code

By Brian Hipple

Experienced BBx® developers, as well as developers with a background in object-oriented design and object-oriented programming, may be a little skeptical when they hear that the latest generation of the BBx language is now 'fully object-oriented' and therefore may have many questions about BBj® custom objects.

This article addresses such questions as "Do BBj custom objects support inheritance, interfaces, encapsulation, polymorphism, access modifiers, and scoping as other object-oriented (OO) languages like Java, C++, or .NET do?" "Is the syntax more comparable to traditional BBx or to these other OO languages?" "What are the benefits of incorporating custom objects?" "What is involved in moving from a procedural coding style to an object oriented style?" And, last of all, "How would a developer integrate custom objects to an existing application or system?"

Object-Oriented Functionality Support

BBj does indeed support traditional OO functionality from inheritance to scoping, giving the application developer the most powerful and versatile development infrastructure.

Picking up the custom object syntax is quite easy for both seasoned BBx developers and experienced OO programmers, as it is a nice blend of BBx and OO type code. Similar to how **DEF FN** and **FNEND** mark the beginning and ending of a function in BBx, the **method** and **methodend** verbs mark the beginning and ending of a method. Traditional OO statements such as **`class`**, **`extends`**, **`interface`**, **`implements`**; and **`private`**, **`protected`**, and **`public`** access modifiers help make the move from other OO languages into BBx an easy transition.

Benefits of Custom Objects

Benefits of incorporating custom objects into an application or system include decreased development time and increased development resources, readability, maintainability, and reusability. Since many colleges and universities teach OO design and OO programming, this is a new widespread opportunity for the BASIS community to tap into the wealth of generally available programming resources. Development time decreases significantly with the new ability to complete code in the BASIS IDE. Code completion provides developers with the ability to view and select methods or objects from a pop-up list. Developers no longer need to open and inspect a source file to determine what functions or routines to call – they just simply select the desired object or method from a popup list. Readability and maintainability of code increases as all related data and functions used to modify this data are contained in custom objects as human readable attributes and methods.

Reusability of code is the major benefit of using custom objects. Developers can write and debug functionality once and reuse it repeatedly via inheritance, encapsulation, and interfaces. For example, a BBj application that uses custom objects will need access to the BBjAPI and should provide the ability to process application events. Creating a BBj Graphical User Interface (GUI) application using custom objects requires access to the BBjAPI as well as access to the BBj SysGui API and the application resource. The classes that represent this functionality appear in **Figure 1**.

The **BBjApp** class has as an attribute, a BBjAPI object (**API!**), which provides access to the BBj API and the method **processEvents** that handles user events for the application. The **BBjGUIApp** class extends the **BBjApp** class to get access to the BBj API and process events functionality, but also has as attributes such as the channel that the sysgui is opened on (**SysGuiChan**), access to the SysGui API (**SysGui!**), and the application resource (**Resource$**). Once the developer writes and debugs these classes, other classes can use these classes and benefit from this proven functionality.

continued...

```
rem BBjApp class definition
class public BBjApp
    field protected BBjAPI API!
rem Default Constructor
    method public BBjApp()
        rem Initialize member variables
        #setAPI(BBjAPI())
    methodend
rem Process events for the BBj application
    method public void processEvents()
        process_events
    methodend
classend

rem BBjGUIApp class definition
class public BBjGUIApp extends BBjApp
    field protected BBjNumber SysGuiChan
    field protected BBjSysGui SysGui!
    field protected BBjString Resource$
rem Default Constructor
    method public BBjGUIApp()
        #super!(); rem Call the super
        rem Initialize member variables
        #setSysGuiChan(unt)
        open(#SysGuiChan)"X0"
        #setSysGui(#getAPI().getSysGui())
    methodend
rem Constructor
    method public BBjGUIApp(BBjString p_resource$)
        #this!(); rem Call default constructor
        #setResource(p_resource$)
    methodend
classend
```

Figure 1. Classes that represent the base for a BBj GUI application

Brian Hipple
QA Test Engineer
Supervisor

Procedural to Object-Oriented

Migrating from a procedural coding style to an OO style involves approaching application development and implementation with a different mindset. Normally, in a procedural application design and implementation, action is performed in a top-down fashion and then refined by adding more details. The task breaks down major functionality into parts comprised of a set of ordered actions. There is usually no relationship, or a very loose one at best, between data and the manipulation of the data. By contrast, in an OO application design and its implementation, action takes place in bottom-up fashion, where individual parts of the task are accomplished in detail and parts are then linked together to form the application. In addition, in OO programming a class is comprised of data called *attributes;* and *behaviors*, also referred to as methods. Being in the same class provides the strongest relationship between data and the manipulation of the data. To understand the references in this article to classes and objects more conceptually, think of "class" as a cookie cutter and the "object" as a cookie.

Migration to Custom Objects

As mentioned earlier, to incorporate custom objects into an existing application, start from the bottom-up. Identify what data the application uses – not just the data on disk, but data structures and groups of related information used in the application. Uniform Modeling Language (UML) object diagrams will now represent custom objects that encompass the application or system. These diagrams are a real help in designing and documenting the application as they describe the custom objects attributes and functionality as well as showing the relationships between the objects. **Figure 2** illustrates design classes with UML diagrams that contain these attributes.

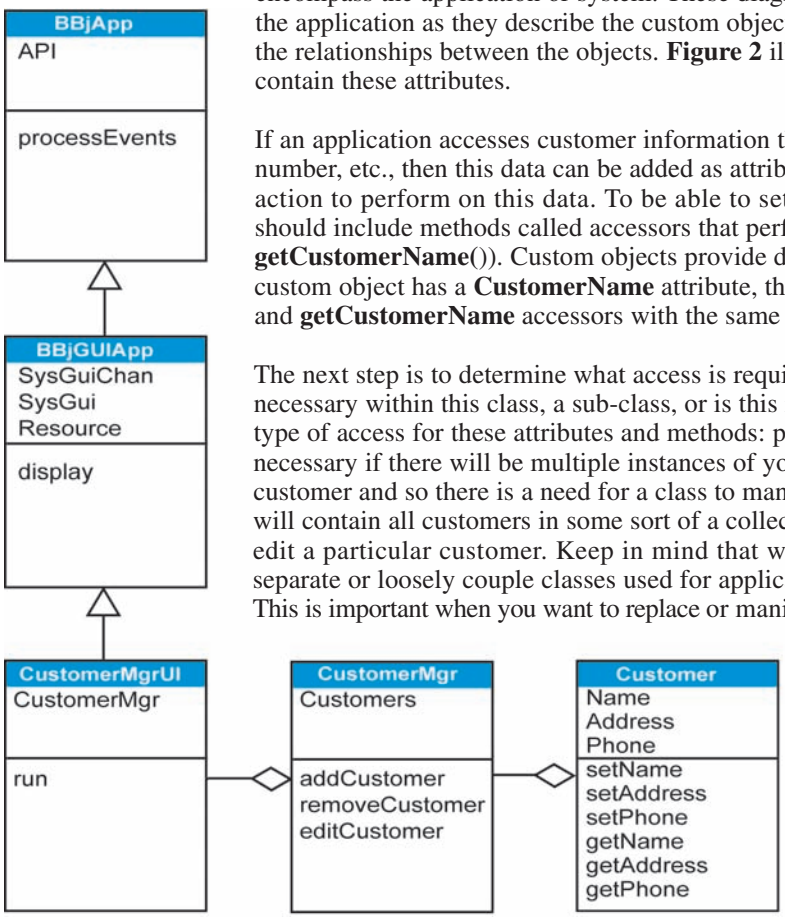


Figure 2. Uniform Modeling Language (UML) object diagram illustrates design class attributes

If an application accesses customer information that includes the customer's name, address, phone number, etc., then this data can be added as attributes of a class (**Customer**). Next, determine what action to perform on this data. To be able to set and get this customer information, this class should include methods called accessors that perform these functions (i.e. **setCustomerName()**, **getCustomerName()**). Custom objects provide default accessors for each class attribute. If the custom object has a **CustomerName** attribute, the class will get, by default, a **setCustomerName** and **getCustomerName** accessors with the same access as the attribute.

The next step is to determine what access is required for these class attributes. Is this data only necessary within this class, a sub-class, or is this information for all? This answer determines the type of access for these attributes and methods: private, protected, or public. Aggregation will be necessary if there will be multiple instances of your data. In this case, there will be more than one customer and so there is a need for a class to manage this information. This class (**CustomerMgr**) will contain all customers in some sort of a collection and provide access to remove, retrieve, and edit a particular customer. Keep in mind that when performing OOD, the developer should separate or loosely couple classes used for application data, user interface (UI), and communication. This is important when you want to replace or manipulate one of these areas without affecting the other. To illustrate this, our sample has a class (**CustomerMgrUI**) that implements the necessary UI functionality for the user to add/remove/edit customer information. However, this class does not directly manipulate the data that contains customer information - it contains the class (**CustomerMgr**) which manipulates the data. Therefore, changing the UI from a CUI to a GUI only changes the UI class (**CustomerMgrUI**) and does not affect the data or access to the data.

Summary

Remember, you do not have to move your whole system to custom objects all at once, you can migrate a piece at a time. BASIS strongly encourages the use of custom objects for your next development project. Developers new to OOD and OOP can find a wealth of books and articles to help in this endeavor. Use UML object diagrams and plan to spend at least 30% of your development time in design. To rework existing code or write new code, take a little time to figure out what your data is, determine how the data is manipulated, and what access is necessary to the data. Following these steps will make your next BBx project using custom objects a successful one!

For more information, read the articles *A Primer for Using BBj Custom Objects* on page 11 in this issue and the e-article *Freedom of Choice: Using Object Code Completion in the IDE* located online at www.basis.com/advantage/mag-v10n1/codecompletion.html

The Scoop on 64-Bit Computing

By Jason Foutz

While new technology can sometimes be agonizing, moving into the world of 64-bit computing does not have to be distressing. In fact, BASIS makes moving your application into this environment quite simple and rather painless.

This article reveals how the hardware in 64-bit computers provides great performance advantages and explores the kind of operating system support that is necessary to use this new hardware. In addition, readers will learn about the necessary tools BASIS provides developers to install and run their application in a 64-bit environment.

The Benefit

In the past, 32-bit processing was more than adequate for most business applications. Since the amount of RAM (Random Access Memory) is the key to an operating system's and application's performance, adding more RAM at today's lower prices has been a sensible way to improve performance; but only up to a point. A 32-bit system can only handle a maximum of 4 GB of RAM, severely limiting the amount of accessible memory in an existing system. If the system runs many applications at once or supports several simultaneous users, it can begin to run more slowly as it runs out of available RAM. Today's modern applications often use cutting-edge features and require even more memory than their predecessors, compounding the problem further.

Manage far more information, faster, and for a higher number of users.

When the computer runs out of available RAM, the operating system compensates by using virtual memory on the disk and swapping memory between programs as needed. Although the operating system has found this available virtual RAM on the disk, the operating system and applications typically slow down to a crawl when using virtual memory. The reason is that the virtual memory supplied by the disk, while plentiful, is thousands of times slower than physical RAM. This is because the disk functions with moving parts, whereas RAM is solid-state technology. While more RAM is not essential for building larger applications, it results in a computer that can manage far more information, faster, and for a higher number of users. Today's 64-bit processors abolish the 4 GB RAM limitation and are capable of addressing up to 18 billion GB of RAM.

Another advantage of 64-bit computers is the ability to perform certain kinds of arithmetic much faster compared to their 32-bit counterparts. A 32-bit computer must break up the addition of large numbers into several parts, whereas a 64-bit computer can perform the operation in a single step. Cryptography deals extensively with these large integers, performing arithmetic on them 64-bits at a time. This nets out to be *more* than twice as fast as 32-bits at a time because it eliminates much of the overhead of breaking numbers down and reassembling them. In addition, audio and video encoding benefit from 64-bit operations since translating a video from one encoding format to another is numerically intensive. Essentially, the processor accomplishes much more per clock cycle with 64-bit operations than it does with 32-bit operations.

The Hardware

Some vendors provided a variety of 64-bit processors for many years. Sun Microsystems developed the SPARC system while IBM used the PowerPC in game consoles and in high-end servers. Various versions of UNIX are available for the various 64-bit processors, but the most prominent processor supported belongs to AMD. They took the 32-bit x86 processor and extended it to a 64-bit architecture, adding more registers while maintaining backwards compatibility. The most attractive aspect of AMD's x64 is its reasonable price and product availability. Since both Intel and AMD are producing 64-bit processors, there is no need to spend millions of dollars on a mainframe for access to an enterprise level processor.

The Software

Taking advantage of the 64-bit processors requires special software. AMD's x64 is compatible with 32-bit programs but 32-bit programs do not take full advantage of the processor's capabilities. Microsoft provides a special 64-bit version, the Windows XP Professional x64 Edition. Though this version of Windows is the only way to run 64-bit code, it is capable of running 32-bit code as well. Linux users must also obtain special versions of their operating system to take advantage of the enhancements of the x64 architecture.

The Support

BASIS helps developers move easily from 32-bit to 64-bit environments, regardless of whether they are using PRO/5® or BBj®. Both PRO/5 version 6.0 and BBj 6.0 are available for several 64-bit operating systems. When installing BBj, developers choose either the 32- or the 64-bit JVM that BBj will use. BASIS supports developers and facilitates deployment by making all of these options available in the BBj installation response file. With BBj, you can even switch between 32- and 64-bit JVMs after installation.

Summary

Clearly, the memory limitation for the 64-bit processor is relatively non-existent for today's application needs and technology. Special versions of Windows and Linux make it possible to take advantage of all the power 64-bit processors can offer. Perhaps more importantly, BASIS supports its developers by giving them control over 32- or 64-bit installations and by running applications identically in either environment. The 64-bit processor really shines in its ability to manage audio and video encoding. As this technology evolves, its ability to pass on these high performance gains will greatly affect day-to-day business applications. The future of 64-bit looks bright.

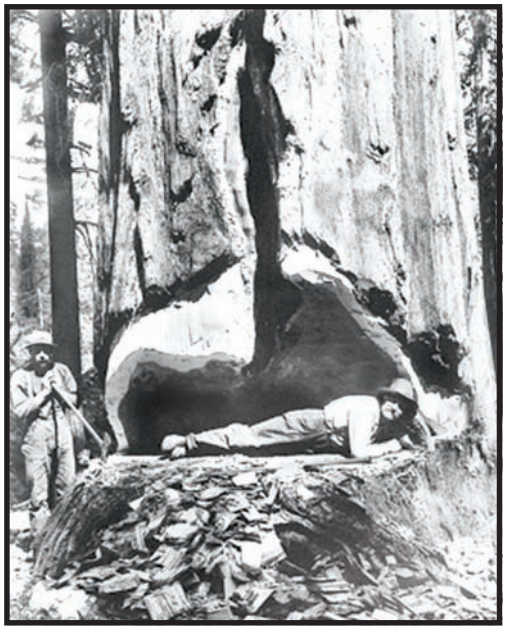
While experts in the computer industry demonstrate repeatedly that predictions about how much memory someone might need in the future are undependable, most will agree that the memory available with 64-bit processing is a safe range, at least for today.

Jason Foutz
Software Programmer

Rooted in Platform-Independence, BBj Supports Borealis Press's Choice of Mac

By Susan Darling, Mike Scully

The Borealis Press produces unique and humorous cards for holidays and special events as well as related journals, refrigerator magnets, note cards, and other specialty gift items. Bookstores, gift shops, stationery stores, and boutiques throughout North America and beyond, distribute Borealis products.



©The Borealis Press, Surry, ME

"The difference between genius and stupidity is that genius has its limits."
— Albert Einstein

Like many businesses with strong graphics and desktop publishing requirements, The Borealis Press is very fond of and highly dependent on the Apple Macintosh computing platform. However, like any business, they still have a need for basic accounting, inventory control, order processing, payroll, and other business management tasks that require financial software. Because the Mac has a limited market share in the general computer world, Borealis found themselves with fewer software choices that met their needs. Like many Apple-centric enterprises, they had to compromise on the features and functionality of the software with which they ran their operations.

A small company, Borealis outgrew the functionality of their business accounting package that was one of the few available for the Macintosh environment. However, as they saw their sales and operations needs expanding rapidly, the software became a large bottleneck in their operations. "*We had some processes that would take what seemed to be an hour to run,*" said Borealis Operations Manager Aimi Baldwin. "*The package used the Mac standard beach ball that would spin around on the screen to show that it was processing. When we needed a report from history, it would be 'beach ball time' and we would leave for lunch.*"

As they shopped for alternatives, Borealis employees kept in mind a unique requirement for their industry: royalties. Like many publishers, they license images (shown above/to the right) and text from various photographers, artists, and authors, and therefore needed to track these for payment and recovery of advances. Along their search, other greeting card publishers recommended their own solution – a blend of Open Systems Accounting Software (OSAS) from TBC International, Inc, a BASIS/OSAS reseller and integration specialist, and an add-on royalty processing suite written by Mike Scully, the owner of Flexible Strategies and a BASIS developer who frequently partners with TBC for customizations.

In conjunction with Richard Paul Thomas of TBC, Mike's ideal solution was OSAS version 7.0 powered by BBj. However, OSAS 7.0 was still in development, and since Borealis needed a quick transition from their Mac-based system, Mike deployed the solution in two phases. In January 2005, Borealis went live with OSAS 6.50 on a Linux server, using their Macs as character terminals. Once the controlled release of OSAS version 7.0 was available, Borealis migrated to the exciting and new graphical, multiplatform, Java-powered version. Since November, Borealis has been running OSAS 7.0 graphically on their six Macintosh desktops (see **Figure 1**), two Windows PCs, and the Linux server, while retaining the characteristic look and feel on each platform.

"*This was my first experience with BBj and honestly, I was a bit out of my comfort zone, but the migration from PRO/5 to BBj was rather seamless,*" says Mike. "*The syntax was familiar, although the move required some client/server considerations, but all in all it was pretty painless. BBj opens the door much wider to new platform opportunities and markets. It uses better syntax, better command line control, and better debugging. What a great way to have GUI and UNIX/Linux too.*"

Mike has plans to put more of the new capabilities of BBj and WebServices to further use. He wants to write a program to run an online address real-time verification whenever a user adds or edits an address. The user then answers a prompt to "accept or reject the verified address." Mike could also use WebServices to display package delivery information directly on the screen rather than their current process that requires a browser.

continued...

Susan Darling
Technical/Marketing Writer

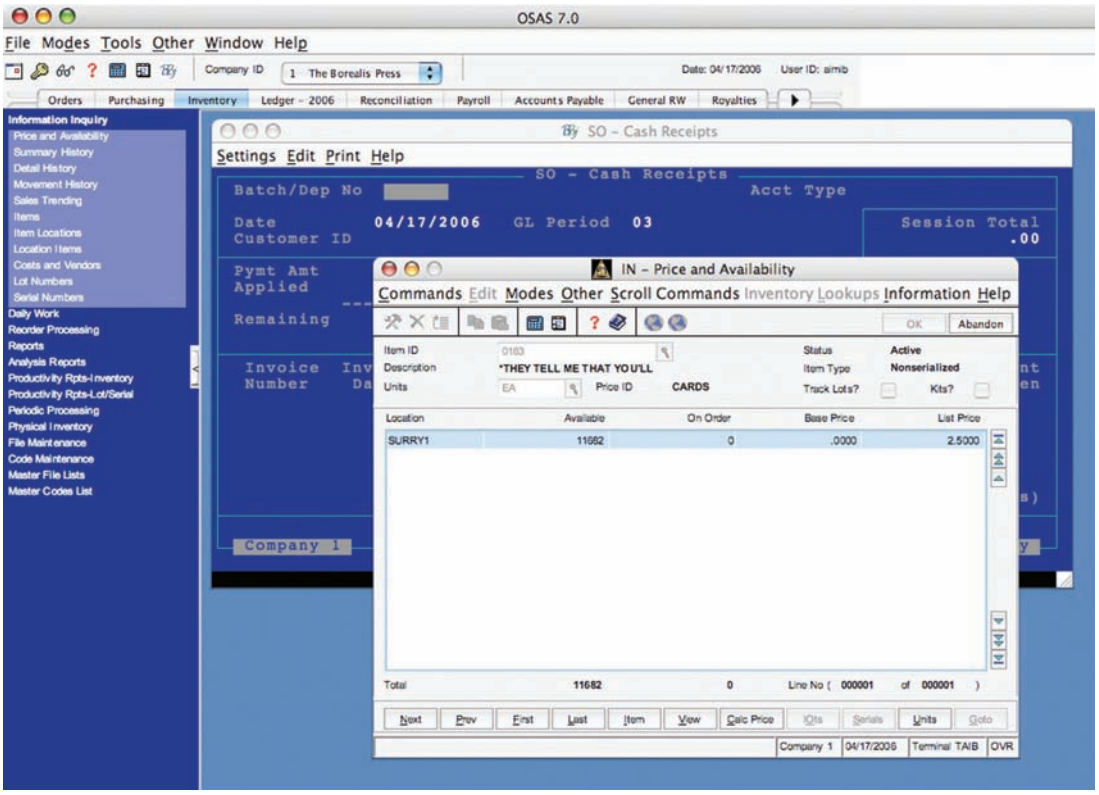


Figure 1. OSAS 7.0 running in GUI and CUI within the MDI on the Apple Mac.

With BBj and Java still so new to so many BBx developers, Mike advises them to get familiar with Java – look for books and other resources to improve their understanding. Brushing up on their understanding of operating system basics and networking fundamentals such as addresses, ports, firewalls, and DNS would help too. Mike adds, "*BBj is so feature-rich, and like Java, fully supports object oriented programming techniques. With tighter security through Enterprise Manager and the option to deploy in mixed CUI and GUI environments, and across different operating systems, BBj is the future.*"

Mike Scully

Mike Scully, founder of Flexible Strategies in Portland, OR, has worked with more than 120 clients across the country in various capacities - custom modifications, needs analysis, hardware implementations, and systems troubleshooting. For over 23 years, Mike has worked with the Open Systems flagship product dating back to their version 1. He has extensive experience in data system integration, hardware, operating systems, LAN and WAN communications, and technical support, with a focus on the needs of distribution, manufacturing, and service companies. He has worked with the BASIS International product suite since BBxProgression/2® and has participated in several TechCon events since 1991.

For more information, visit Open Systems, Inc. at www.osas.com and The Borealis Press Inc. at www.borealispress.net

Solving the Locked Record 'Whodunit'

The Situation

By Susan Darling

Colonel Mustard calls the operations room in a panic. He must access a critical intelligence master record on the computer database immediately but cannot; the record is 'busy.' Quite obviously, the record is locked but who "dunit" and how do we "un-dunit"?

In the past, identifying the locked-record perpetrator was difficult. Today's tools will ID which user needs to exit the record, but what if that user's office is locked? What if it is at a remote location somewhere else in the world?

Earlier that day, Mr. White made several mission critical updates to this same intelligence master record. Without exiting that record, he left for his usual lunch. Normally, resolution would be just an hour away, but this day an offsite emergency has tied Mr. White up the entire afternoon. Furthermore, Mr. White has been accessing the system from a remote location several time zones away.

The Solution

By Jim Douglas

While national security may not really be threatened by the locked record illustrated in this tongue-in-cheek introduction, a locked customer record could result in an equally vulnerable and costly situation. In PRO/5® on UNIX, the only option has been something like this:

```
$ /sbin/fuser INVENTORY
INVENTORY:            10033
$ ps -p 10033 -f
UID      PID  PPID  C STIME TTY       TIME CMD
jsmith 10033 15419  0 15:08 pts/1 00:00:00 /usr/local/bin/pro5/pro5s -m1024 -cconfig.bbx
$ kill -9 10033
```

A more direct approach is the **`fuser -k`** option that kills all processes that have opened the specified file:

```
$ /sbin/fuser -k INVENTORY
INVENTORY:            10033
```

While this approach can work, it has some significant problems.

- It does not work with BBj®
- It only works on UNIX (and UNIX-like) systems and it requires the fuser utility
- It identifies all users who have the file open, but not the user who is locking a particular record, which is a problem if multiple users have the same file open (the typical case)
- Killing the process could cause collateral damage to other tasks the locked user was performing so the ability to force-close the channel that has the record extracted would be less drastic

The BBj Enterprise Manager includes a much more flexible way to view open files and to optionally force a file closed. To use this feature, open BBj Enterprise Manager and click on Open Files as shown in **Figure 1**.

continued...

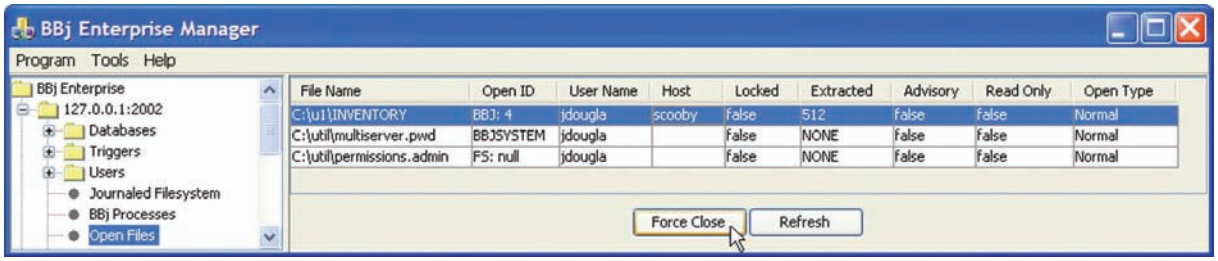


Figure 1. Open Files dialog box

Jim Douglas
Software Engineer
Contractor

Enterprise Manager is an improvement over the UNIX fuser approach in a few ways.

- It works on all BBj platforms, not just UNIX, and it does not depend on an external utility program
- It shows which file references are currently extracting a record (as opposed to merely having the file open)

But this approach still has some problems.

- The user must have BBj administrator privileges to log in to Enterprise Manager
- The user must navigate through a very technical screen and identify the correct file channel from the list, which can be very difficult, especially if there are dozens or hundreds of files opened across the system
- If multiple channels are currently extracting a record, there is no way to tell which one is holding the particular record of interest

BBj 6.0 improves on this functionality in the following ways.

- The new BBjOpenFileInfo API exposes information about open files in a format that can be processed so end users do not need to go into Enterprise Manager
- The TCB(10) function returns the lockbyte within the file when reporting !ERROR=0, enabling developers to locate the specific file channel that holds the lock to the record they are currently attempting to read

First, run **`program1.bbj`** in BBj (**Figure 2**).

```
0010 filename$="sample.dat"
0020 erase filename$,err=*next
0030 mkeyed filename$,1,0,16
0040 open (1)filename$
0050 write (1,key="X")"X"
0060 extract (1,key="X")
0070 escape
0080 print fid(1)(9)
```

Figure 2. Code sample **`program1.bbj`**

Next, with that program sitting at the escape, run **`program2.bbj`** in a different BBj 6.0 (**Figure 3**) interpreter session.

```
0010 filename$="sample.dat"
0020 open (1)filename$
0030 extract (1,key="X",err=errtrap)
0040 print "Record successfully extracted!"
0050 stop
0060 errtrap:
0070 if err<>0 then escape
0080 channel=tcb(12)
0090 lockbyte=tcb(10)
0100 fs!=bbjapi().getFileSystem()
0110 fileinfo!=fs!.getFileInfo(channel)
0120 filename$=fileinfo!.getFilename()
0130 filename$=fnslash$(filename$)
0140 BBjAdmin!=bbjapi().getAdmin("admin","admin123")
0150 BBjVector!=BBjAdmin!.getOpenFileInfos()
0160 for i=0 to BBjVector!.size()-1
0170    BBjOpenFileInfo!=BBjVector!.get(i)
0180    openfile$=BBjOpenFileInfo!.getFilename()
0190    openfile$=fnslash$(openfile$)
0200    found=openfile$=filename$ and BBjOpenFileInfo!.getExtracted()=lockbyte
0210    if found then break
0220 next i
0230 if found then print openfile$," closed"; BBjOpenFileInfo!.forceClose()
0240 retry
0250 def fnslash$(x$)
0260    x=pos("\"=x$)
0270    while x
0280       x$(x,1)="/",x=pos("\"=x$)
0290    wend
0300    return x$
0310 fnend
```

Figure 3. Code sample **`program2.bbj`** for finding who locked the record

continued...

Now type RUN at the first interpreter session. Attempting to access the file channel now reports !ERROR=13 because the file has been forced closed.

```
READY
>run
0070 escape

READY
>run
!ERROR=13  (File is closed.)
0080 print fid(1)(9)

READY
>
```

The previous sample contains a lot of activity. **Figure 4** shows a line-by-line breakdown of the second program.

Lines	Notes
0010..0050	Open a file, attempt to extract a record.
0060..0240	Error handler – on !ERROR=0, force close the channel that holds the lock.
0070	Only proceed if this was !ERROR=0 (locked record).
0080	Retrieve the last channel referenced (the channel with the error).
0090	Retrieve the reported lockbyte in the file.
0100..0130	Get the name of the file opened on this channel.
0140	Get the BBjAdmin object that is needed for any tasks that require BBj Administrator privileges.
0150	Get the list of all files currently opened within this copy of BBjServices.
0160..0220	Loop through the list of opened files and find the one that holds the lock that just reported !ERROR=0. The channel can be identified by matching on both filename and lockbyte.
0230	If the lock-holder was successfully identified, print it and force-close the channel.
0240	Retry the extract. With the lock-holder closed, it should succeed this time.
0250..0310	Utility function to convert any back-slashes in a filename to forward slashes (used for comparing filenames).

Figure 4. A line-by-line breakdown of the activity in the second program

Summary

Today, BBjOpenFileInfo solves the longstanding locked-record mystery for interpreters running in the same virtual machine as the DBMS and provides an easy solution to who "dunit" and how to "un-dunit" scenario. Locked customer records will no longer be a difficult issue to deal with in day-to-day operations.

Download the sample programs referenced in this article from www.basis.com/advantage/mag_v10n1/lockedrecord.zip

For a more complete and polished sample, download a mockup of a customer master file maintenance program from www.basis.com/advantage/mag-v10n1/customer.zip

Visual PRO/5 6.0 Gives Apps an XP or Vista Contemporary Look and Feel

By Nick Decker

At first glance, an operating system's theme may appear to be just another pretty face. However, this look and feel is more than skin deep – it defines how users interact with the operating system and its applications. End users become accustomed to the target look and feel (L&F), which provides comfort and familiarity. It makes applications more intuitive as certain visual styles can effectively relay at a glance the use and capability of a particular control.

Until recently, Visual PRO/5® applications were not able to take on the target operating system's L&F, making it appear antiquated and less intuitive to use. Visual PRO/5 6.0 changes all of that, automatically presenting any BBx® program in the modern L&F of the target operating system. No matter if the application is running on a current operating system, such as Windows XP, or an operating system of tomorrow such as Windows Vista, the application runs in the best possible light and fits right in with the rest of the operating system.

Visual PRO/5 and Themes

BBj® has supported themes from both Java and SkinLF for years. But what about themes for Visual PRO/5? Until now, the only way that developers could influence the L&F of their Visual PRO/5-based application was to add a raised or sunken border to a control. The client edge and raised edge options gave the GUI designers the ability to mimic the L&F of the time – the Windows 95 3D beveled L&F. There are a couple of very important distinctions about how BASIS implemented this in earlier versions of Visual PRO/5.

There is more!

Read this complete article online

for a closer look at skinning's beginnings, the impact of themes across other industries, and the progression to today's modern look and feel now available in Visual PRO/5 6.0. Go to

www.basis.com/advantage/mag-v10n1/supp.html

Firstly, the developer was in charge of determining which edge options the controls should have. That put the onus on the BASIS developers to create an application that adhered to the L&F. In other words, the L&F did not come through automatically – the developer essentially 'hard-coded' the appearance for each control. While this did not seem so egregious at the time since there was only one L&F, most programmers today shudder at the thought of hard coding anything, believing that hard coding is bound to get you into trouble sooner or later. As it turns out, hard coding the L&F was no exception.

Secondly, the client and raised edges were available to every control. That is a significant point as the Windows 95 L&F dictated that some controls had a beveled edge while others did not. Since Visual PRO/5 would add or remove edges without discernment, it enabled developers to easily bypass the conventions of the L&F and create non-standard GUIs. This was non-standard in the sense that Visual PRO/5 applications could misuse existing elements of the L&F, sometimes resulting in an odd and awkward looking interface that appeared out of place. This was not a good feature as it allowed the developer to introduce anomalies to the L&F that deviated from the norm. **Figure 1** demonstrates the effect, showing various combinations of client and raised edges of the edit and button controls.

To Visual PRO/5's credit, it gave the correct defaults to the controls. In the example in **Figure 1** the edit control has the client edge property 'on' by default while the button has no edges defined. Looking at this screenshot, the controls appear correct when rendered with these options; however they do not look correct when rendered with other options. Specifically, the edit control looks normal with the client edge, having the familiar Windows 95 sunken beveled look. When created with a raised edge, however, it 'pops out' of the form and looks more like a white button than an edit control. Likewise, the button looks correct without any edges. Adding a raised edge to it makes it look like its sitting on top of yet another button. If the developer modified the default edges for controls, the resultant application ended up looking different from other Windows applications and could, in some cases, present a confusing interface to the user.

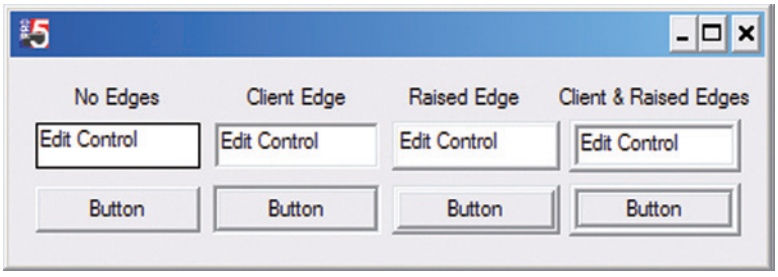


Figure 1. Visual PRO/5 and the various combinations of client and raised edges on controls

Nick Decker
Engineering Supervisor

continued...

Visual PRO/5 6.0 Adds Support for Themes

Visual PRO/5 programs, regardless of which version of Windows they ran on, always displayed in the 'hard coded' Windows 95 L&F. As far as the 'look' goes, a Visual PRO/5 application always used the Windows 95 beveled edges, even though Microsoft eliminated them in Windows XP and the upcoming Vista. In addition to presenting the wrong look for the operating system, the 'feel' part of the equation was missing as well. The hover effects, for example, were conspicuously absent. Today's end users are now accustomed to the feedback provided by the operating system in subtle ways – fade and glow effects that result from hovering over a control, the change in a button's appearance when it is the default button on the form or when it currently has focus, and so on. None of these effects were available because Visual PRO/5 did not automatically adapt to the current L&F.

Screenshots in the following figures demonstrate this concept using the button control. **Figure 2** shows what the button control looks like in the five possible states when theme support is disabled. In addition to having the older Windows 95 look, the 'feel' aspect is missing since the hovered state is the same as the normal state. In other words, when the user moves the mouse over the button, nothing happens.

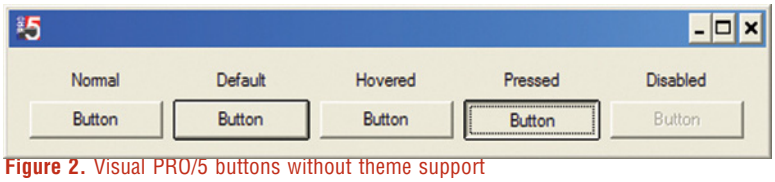


Figure 2. Visual PRO/5 buttons without theme support

Contrast this with **Figure 3** that shows the button controls in the same five states but themed in Windows XP. The look of the controls is very different, displaying round-cornered buttons with a slight gradient to add depth and colors to indicate state. The feel is very different as well – when

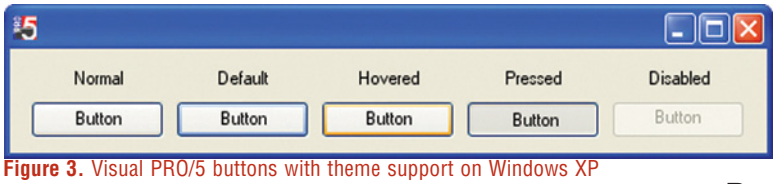


Figure 3. Visual PRO/5 buttons with theme support on Windows XP

moving the mouse over the button, the button changes its appearance to the hovered state of glowing orange. Taking the 'feel' aspect one step further in Vista (**Figure 4**),

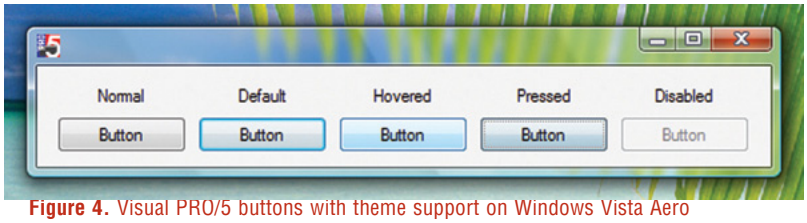


Figure 4. Visual PRO/5 buttons with theme support on Windows Vista Aero

the theme for a default button goes beyond a simple blue outline and actually pulsates, changing color in real time.

These effects are not just limited to buttons; they affect many of the other controls, such as the checkbox control. If Visual PRO/5 uses the old Windows 95 L&F shown in **Figure 5**, there would be only two possible states for a box – checked and unchecked. Furthermore, the appearance of the checkbox does not change when hovered or pressed. Running the same program under Windows XP or Windows Vista with theme support increases the number of states from two to six as illustrated in **Figures 6 and 7**; checked, unchecked, hovered checked and unchecked, and pressed checked and unchecked.

Automatic Theme Support

Visual PRO/5 6.0 solves the theme support problems of past revisions and introduces full-scale theme support. The first and probably most important feature of Visual PRO/5's theme support is that it now implements automatically. BASIS developers do not have to change any code or modify any resources. Their applications will magically take on the L&F of its operating system's theme. In other words, any legacy Visual PRO/5 application will run in Visual PRO/5 6.0 under three different operating systems – Windows 2000, Windows XP, and Windows Vista – and will display the user interface consistent with the L&F of the operating system. Not only will all of the controls display correctly with the appropriate border styles and so on, they will also behave correctly by exhibiting hover effects, rendering the default button state, etc.

The second feature of Visual PRO/5's theme support is that of normalizing the appearance of controls with respect to their edges. As mentioned before, previous versions of Visual PRO/5 allowed developers to create non-standard controls by mixing and matching various edge styles. In order to comply fully with the target L&F, Visual PRO/5 6.0 would have to solve the problem of non-standard edges. Previous versions merely suggested that developers provide a consistent L&F by presenting suitable defaults. Visual PRO/5 6.0 had to be much more stringent than past versions to accomplish its ambitious goal so BASIS designed it to prohibit edges that failed to comply with the target L&F. As a result, it is possible to upgrade legacy applications instantly to a modern L&F without requiring any developer effort.

Backwards Compatibility

Normalizing the controls in this manner can have an unexpected side effect. For example, if a developer defines a control not to have either a client edge or a raised edge, but the target theme dictates that the control must have an edge, Visual PRO/5 will adjust the control accordingly to draw the appropriate edge. In most cases, the newly added edge appears as a few pixels on all sides of the control – seemingly hardly enough of a change to make a noticeable difference. However, a four-pixel adjustment in the control's width and height must come from somewhere. The two places where this adjustment may occur are 1) outside of the control or the *non-client* area or 2) inside of the control or the *client* area. Visual PRO/5 makes the adjustment via the

continued...

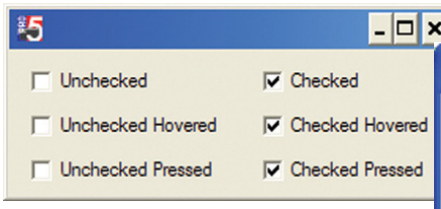


Figure 5. Unthemed checkbox controls

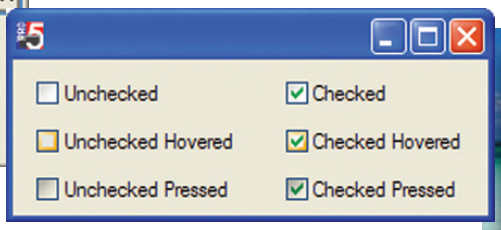


Figure 6. Themed checkbox controls on Windows XP

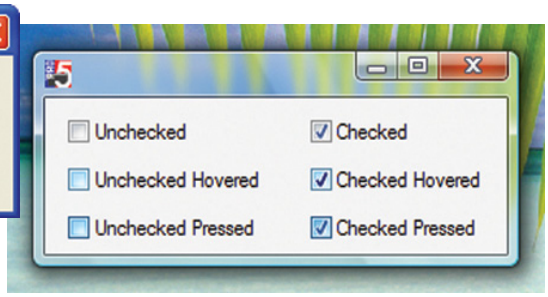


Figure 7. Themed checkbox controls on Windows Vista Aero

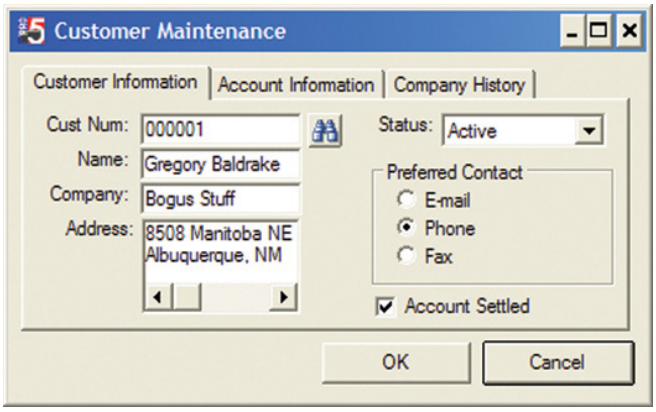


Figure 8. Sample application running non-themed

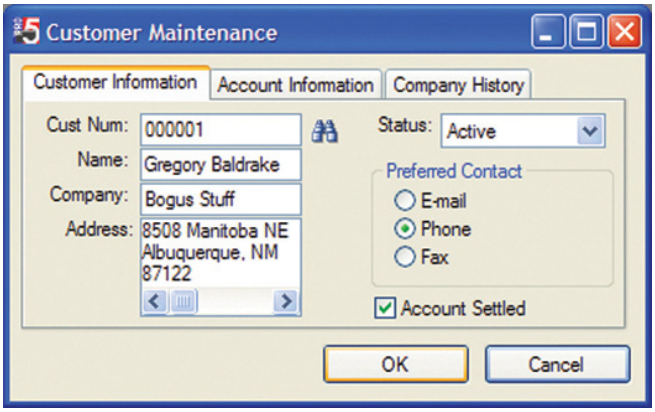


Figure 9. Sample application running with Windows XP Luna

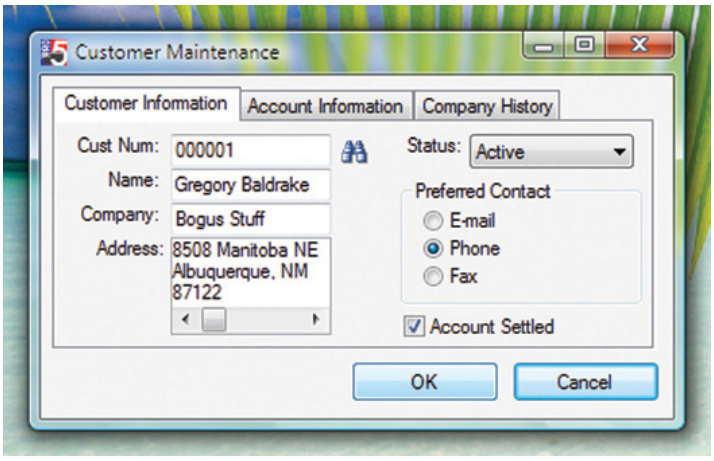


Figure 10. Sample application running with Windows Vista Aero

latter option – decreasing the size of the control's client area in order to make room for the new edge. Option 2 is generally preferred, as the alternative would increase the size of the controls, possibly resulting in the controls overlapping one another on the form if they originally appeared in close proximity. Taking the space from the client area of the control prevents them from growing larger and reduces the number of possible harmful side effects.

However, option 2 is not without potential side effects since the content of the control now has a smaller container. The grid is a good example of this problem as it is an amalgamation of three separate controls – the grid itself, the row header, and the column header. Older versions of Visual PRO/5 strictly applied the edges to the grid control itself and completely ignored the headers. This led to a sort of disjointed control as the headers appeared to be floating separately from the grid. Most modern L&Fs apply the edge to the control as a whole – the grid and the two headers. Visual PRO/5 6.0 does this too when themes are turned on, but this leads to a discrepancy in how the grid is rendered compared to previous versions of Visual PRO/5. The result is that the grid's client area may vary slightly with theme support turned on. In most instances, applications will not be susceptible to that minute difference. However, in case this does affect an application, BASIS created a keyword to appear in the configuration file to ensure backwards compatibility. Simply add the keyword **DISABLE_GRID_BORDER_THEME** to the **config.bbx** file to instruct Visual PRO/5 to enable all theme support except for grid borders. Other parts of the grid, such as the headers, will still be themed appropriately, but the edges will remain consistent with past versions to ensure correct sizing and placement of the grid control.

No doubt, most developers will love Visual PRO/5's ability to change like a chameleon, automatically taking on the L&F of the operating system on which it is running. In the special case that requires the L&F of previous versions, a developer can bypass theme support all together by adding yet another new keyword, **DISABLE_THEME_SUPPORT**, to the **config.bbx file**. This forces Visual PRO/5 to revert to the legacy behavior, drawing all of the controls in the older style with the user-defined edges as shown in **Figure 8**.

Summary

Contemporary operating systems utilize themes to define the L&F of the operating system and its applications, customizing and enriching the user experience with specialized graphics and effects. BBx developers can now take advantage of these themes by running their applications in Visual PRO/5 6.0 which fully embraces the capabilities of the current L&F. No matter if the application is running on a current operating system such as Windows XP (**Figure 9**), an operating system of tomorrow such as Windows Vista (see **Figure 10**), or even

continued...

on an operating system utilizing third party themes (see **Figure 11**), the application runs in the best possible light and fits in with the rest of the operating system.

Figure 11. Sample application running a variety of third party themes

Download Visual PRO/5 6.0 today and try out the contemporary L&F. Go to www.basis.com/products/downloads.html

Read about the journey of themes from the first skins to customization across many industries in the e-Supplement at www.basis.com/advantage/mag-v10n1/supp.html

For additional information related to themes and skins, read *BBj 2.0: More Than One Way To Skin An App* at www.basis.com/advantage/mag-v6n1/skin.html

Type Checking With "bbjcpl"

By David Wallwork

The traditional BBx® language has only three variable types; string (**A$**), number (**A**) and integer (**A%**). Since the introduction of embedded Java, BBj® gave the BBx developer the ability to work with Java objects as well as these three native BBx types. In the 6.0 release of BBj, the available variable types are even broader with the introduction of custom objects; objects defined completely in BBj program files.

While using Java objects and BBj custom objects allows BBj developers to write sophisticated object-oriented applications, these same object-oriented applications have the potential for hard-to-find type check errors. BBj 6.0 provides an easy-to-use tool for discovering these errors – new type checking options in the **bbjcpl** executable that compiles ASCII program files into binary tokenized programs. In this article, we will examine a small program that demonstrates some of the errors that may occur when using object-oriented code, and then show how to discover these errors using **bbjcpl** with these new options.

Consider the program in **Figure 1**. This program contains problems that will generate runtime errors when executed.

```
0010  rem  demo1.bbj

0200  A! = "abc"
0210  B! = 5
0220  print A! + B! ; rem runtime error

0300  C! = "abc"
0310  D! = 5
0320  print C! + D! ;  rem runtime error

0410  x! = bbjapi()
0420  x!.getExistingNamespace( 44 ) ;  rem runtime error
```

Figure 1. Code sample that produces runtime errors

Problem 1

At lines 200-220, this program assigns a string value to **A!** and a numeric value to **B!**, and then attempts to add **A!** to **B!**. Line 220 will generate a runtime error because BBj does not support adding a numeric value to a string value.

Problem 2

The same problem occurs again at lines 300-320. The problem appears twice for demonstration purposes. Below we will resolve the problem at line 200-220 differently from the resolution offered to the problem at line 300-320.

Problem 3

At line 420 the program attempts to pass a numeric value to the method **getExistingNamespace()**. Because that method accepts a string parameter rather than a numeric parameter, the program will generate a runtime error when line 420 is executed.

The entire program is syntactically correct. This means it will load and list correctly and none of the lines will be marked as syntax errors. And yet, we can see from looking at the program that these statements will cause runtime errors. One might ask, "Why is the compiler not able to recognize these problems and mark them as errors during compilation?"

David Wallwork
Senior Architect

The reason that a human reading the program can recognize the errors is that the reader has additional information about the variables. This additional information is *type information*. For example, the user understands from looking at line 200-210 that variable **A!** is of type "string" and the variable **B!** is of type "number." And the user can recognize that at line 220 the program is attempting to add a number to a string. Similarly, at line 420 the reader knows (or can discover) that the method **getExistingNamespace()** requires a string but that what is being passed is a number.

continued...

If there were a way to provide type information to the compiler, then the compiler would also be able to recognize these problems. In BBj 6.0, **declare** statements provide this type information. The program in **Figure 2** uses **declare** statements to provide this information to the compiler.

```
0010  rem demo2.bbj

0100  declare BBjNumber A!
0110  declare BBjNumber B!

0200  A! = "abc"
0210  B! = 5
0220  print A! + B! ; rem runtime error

0300  C! = "abc"
0310  D! = 5
0320  print C! + D! ;  rem runtime error

0400  declare BBjAPI  X!
0410  X! = bbjapi()
0420  X!.getExistingNamespace( 44 ) ;  rem runtime error
```

Figure 2. Declare statement code sample

By adding the **declare** statements at line 100, 110, and 400, we are telling the compiler what type we want variables **A!**, **B!** and **X!** to represent in the program. This allows the compiler to recognize some of the problems that are in our program. Because we have not declared the type for the variables **C!** and **D!**, the compiler still cannot recognize the problem at line 300-320. If we run the compiler on this program using the **-t** option that enables type checking, we receive the output shown in **Figure 3**.

```
> bbjcpl -t demo2.bbj

demo2.bbj: type check error [Incompatible assignment from <BBjString> to
<BBjNumber>] at line 200 (6): A! = "abc"

demo2.bbj: type check error [No match for method
com.basis.bbj.proxies.BBjAPI.getExistingNamespace(<BBjInt>)] at line 420 (16):
x!.getExistingNamespace( 44 ) ;   rem runtime error
```

Figure 3. Type checking output using the -t option

This output tells us that at line 200 we have mistakenly assigned a numeric value to a variable that is meant to be a string variable and that at line 420, we have attempted to invoke a method that does not exist.

Also notice this output still does not give any information about our problem at line 300-320 because the compiler has no type information about the variables **C!** and **D!**. If we run the compiler with the **-t -W** options that enable type checking and related warning messages, it will list all the lines that have type check errors. In addition, it will also list all the lines that may have errors but could not be checked because there is no type information for their variables. Using the **-t -W** options, we receive the output displayed in **Figure 4**.

continued...

```
> bbjcpl -t -W demo2.bbj

demo2.bbj: type check error [Incompatible assignment from <BBjString> to
<BBjNumber>] at line 200 (6): A! = "abc"

demo2.bbj: type check warning [Undeclared variable: C!] at line 300 (10): C! ="abc"

demo2.bbj: type check warning [Undeclared variable: D!] at line 310 (11): D! = 5

demo2.bbj: type check warning [Undeclared variable: C!; Undeclared variable:
D!] at line 320 (12): print C! + D! ;  rem runtime error

demo2.bbj: type check error [No match for method
com.basis.bbj.proxies.BBjAPI.getExistingNamespace(<BBjInt>)] at line 420 (16):
x!.getExistingNamespace( 44 ) ;  rem runtime error
```

Figure 4. Type checking output using the -t -W options

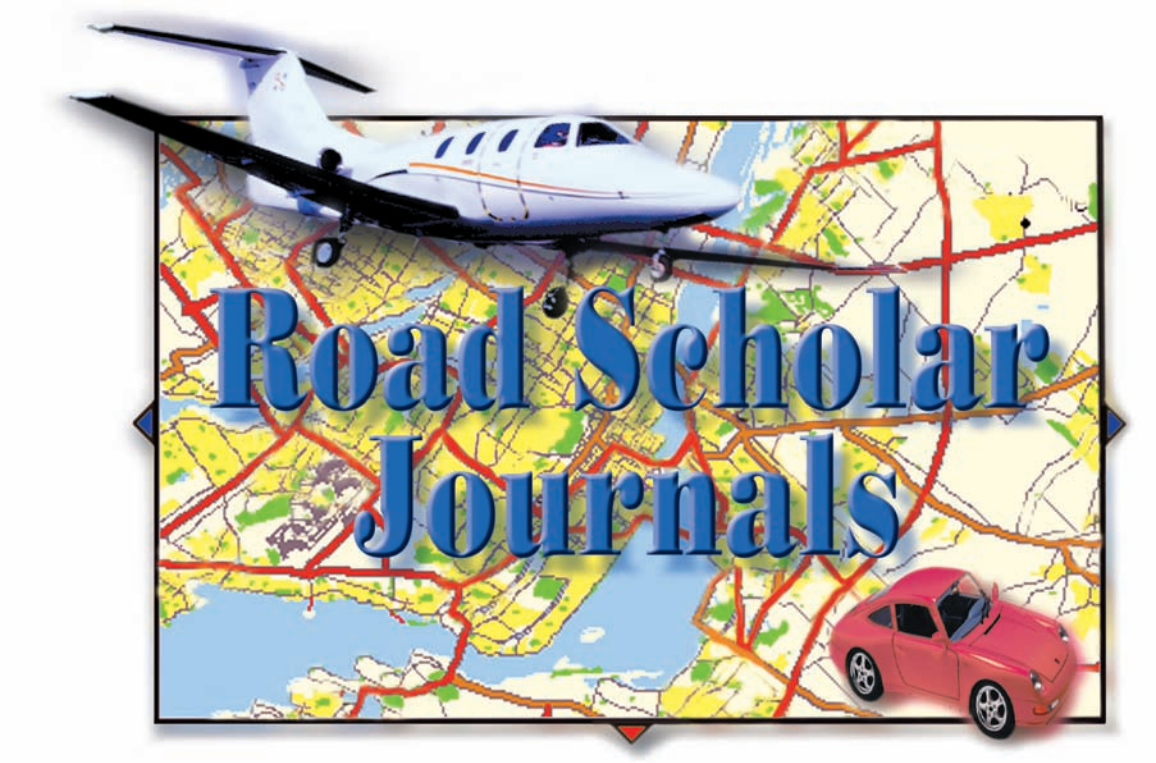
This output tells us about the errors on line 200 and line 420, and warns us that lines 300-320 use variables that have not been declared and so could not be checked for errors.

Summary

Using the **`declare`** verb along with the type check options of BBj 6.0 allows the developer to discover many programming problems at compile time which, if left undiscovered, would become runtime errors. Compile time errors are vastly preferred since they appear during development and provide an opportunity for correction before deploying the application. Runtime errors, on the other hand, occur during execution of the program – oftentimes in a production environment – and therefore, tend to be more costly and difficult to track down.

For a complete discussion of type checking errors and the new options for bbjcpl, refer to the *Type Checker Overview* at www.basis.com/solutions/TypeChecker.pdf

For more information about BBj custom objects see *A Primer for Using BBj Custom Objects* article on page 11.



TechCon2006 in Las Vegas Sizzles

Las Vegas is known around the world for two things - gambling and hot temperatures. Well, the temperature was not the *only* hot thing in Vegas the first week in May. BASIS presented TechCon2006 on May 7-9 at the Marriott Renaissance Las Vegas. While playing the slots is always a gamble, the sure bet was that BBj® 6.0 is the most advanced, feature-rich product BASIS has ever released.

The focus for TechCon2006 was the new BBj 6.0 features and functionality. The team of presenters included Nico Spence, Dr. Kevin King, Brian Hipple, Nick Decker, and Jon Bradley. Together, they debuted the latest features and functionality in the BASIS products - custom objects, VKEYED and SQL file types, new 64-bit ports, Mac OS/X support, triggers and stored procedures, FormBuilder and AppBuilder.

While we received great reviews for the technical content of the general sessions, TechCon2006 was more than "all work and no play." BASIS hosted evening events with excellent food and the top-notch entertainment one expects

continued...

Laurence Guiney
Senior Account Manager

General session at TechCon2006

Monday evening's entertainment

Brian Hipple gives an overview of the BASIS IDE to an interested group of BBx developers during post-TechCon2006 training

in Las Vegas. On Sunday evening, attendees and their companions experienced a "Vegas Warm-up" with cocktails and a delicious buffet. Brian Campbell and his wife Norann strolled through the crowd sharing amazing magic tricks and demonstrating mind-reading skills, leaving several people scratching their heads and saying "how did they do that!?"

After a delicious sample of the "Taste of Vegas" on Monday evening, TechCon attendees, exhibitors, and BASIS employees enjoyed the musical impressions of Robbie Howard and his band. The group performed classic hits from a variety of eras and styles from Willie Nelson to Air Supply to Tom Jones and Julio Iglesias. Their humor and music was thoroughly entertaining and the consensus was that the night ended too quickly.

Following TechCon2006, attendees took the opportunity to sharpen their skills at a variety of one-day training courses held over three days. Many attendees registered for multiple courses to take advantage of the expertise of the instructors and further network with other developers. This post-TechCon training was a great success, filling most classrooms nearly to capacity.

While the saying goes "what happens in Vegas, stays in Vegas," TechCon2006 attendees left with exciting new technology to share with their customers and several fond memories. If the two months that followed TechCon2006 are any indication of the response, numerous attendees are taking home and beginning to use the hottest BBx® technology yet.

— Laurence Guiney

OSAS Conference is Hot

On June 8-10, Nico Spence, Dr. Kevin King, and I attended the Open Systems resellers conference held in beautiful downtown Minneapolis. Generally, the weather is very pleasant this time of year but it seemed unseasonably warm. Sure enough, Minneapolis was having a heat wave after a mild winter (mild in Minnesota terms!).

To introduce this year's conference theme "Partners for Profit," Michael Bertini, Open Systems, Inc. (OSI) CEO, shared his ideas on global opportunities, technology changes and challenges, and the importance of partnership. Modeling such a partnership relationship, BASIS and OSI worked closely together to launch OSAS Version 7.0 running BBj®. This joint venture promises to be a profitable venture for OSI. Paul Lundquist, Vice President of Sales, said that 57% of the fulfillments have upgraded to OSAS 7.0.

The OSAS reseller channel is gearing up for the many opportunities to follow this latest release and the thousands of existing users that are waiting to take advantage of it. To meet these many challenges, OSI provided pre-conference training for OSAS 7.0 and BBj. In addition, Nico and Kevin conducted breakout sessions for new BBj language features and DBMS features to educate the channel further. Class attendees were pleased to see where BASIS is heading and excitement filled the room.

BASIS was among 12 other exhibitors to display their technology. It was great to see a room full of exhibitors and experience all the networking that took place among the attendees - talking to old friends and making new contacts.

As always, OSI conducted a well-organized conference with a wealth of breakout sessions, networking opportunities, and fun events. BASIS congratulates OSI on another successful conference and says, "Thank you," for the opportunity to be part of it. We are committed to our partner Open Systems, Inc., and to the success of OSAS 7.0 and beyond.

— Gale Robledo

Gale Robledo
Account Manager

European End Users Meet in Germany

The weather was warm and sunny for the first European BASIS End User Seminar on June 20-21, 2006 in Germany's famous Black Forest. Karl Knauer KG, a company of over 400 employees and a Business BASIC user for decades (see inset), hosted this event. As host, they had the opportunity to show their own BBj programming successes with their 160 concurrent user system and meet with BASIS staff and other end-user companies.

Over 20 programmers and executives from Germany, France, Switzerland, and Belgium attended this event – the first of its kind for many years – in order to network with other end users and to validate their software development strategic choice of BBj technology. In Knauer, BASIS found a perfect host for this purpose, while surrounded by beautiful and interesting scenery.

The convention was a success for all. BASIS learned more about the needs and requirements of their self programming end user customers, which is sometimes quite different from software development companies. In turn, the attendees gained comprehensive insight into the options that BASIS offers with BBj and how Knauer is taking advantage of these capabilities. For Knauer, they received important feedback and confirmation about their progressive development strategy with BBj.

Given the success of this event, BASIS plans to hold these sessions annually. We invite companies interested in showcasing their application and hosting next year's seminar to contact us.

— Stephan Wald

Karl Knauer KG, lead by executive manager Joachim Würz and IT manager Peter Treier, is a customer of BASIS Europe Professional Services, the new business unit of BASIS Europe. This new unit offers programming, training, and support to BASIS partners, resellers, and end-user companies with their own IT development staff. Knauer operates three factories in Germany and Poland that produce packaging goods and advertising material for worldwide customers including some Fortune 100 companies. BBj is Knauer's key tool for their data processing – accounting, sales, productions and logistics, with a centralized system connecting all three sites. Currently they are running BBj 5.031 on their 160-user system with plans to upgrade to 6.0 in October 2006. Their application mixes an "old" character-based application with some new, highly integrated graphical programs. All new development supports the full feature set of BBj without losing the code base of the existing, proven solution with its /M/A/I origins. www.karlknauer.de/index_yf.html

Stephan Wald
Germany

BASIS Tours Europe With New Partnership Program

By Johannes Fritz

In January of this year, Nico Spence, Herbert Schmitz (still a great expert on German BASIS customers), Sandrine Eustace, and I set out on a journey from north to south Germany to present the newly released BASIS Partnership Program to our most valuable customers.

For the initial presentation, we chose the favorite high-tech city of Hanover, well-known for hosting the annual CeBIT trade show. Many excited customers arrived early so that they would not miss any of the interesting presentations and news that led to a very relaxed but still intense conversation among Partnership customers and the BASIS team.

After the friendly warm-up, Herbert presented the cornerstones of the new BASIS Partnership Program, explaining how BASIS intends to increase the successful partnership together with their customers for the future. The current model that evaluates customers by their annual sales volume has become somewhat outdated.

In contrast to this, BASIS developed the new Partnership Program in cooperation with many customers and takes customer needs into consideration to provide the following demand-oriented components:

- Ordering via the BASIS b-commerce® system
- BASIS product know-how
- Sales and marketing plan
- SAM plan
- Sales volume

This arrangement allows for a far more specific support of customer needs than currently possible, resulting in the new classifications of Bronze, Silver, Gold, and Platinum Partners.

The subsequent discussion showed that the long-term preparation of the new Partnership Program by the BASIS team in conjunction with intensive talks with their customers was perfectly tailored to the needs of customers and was well received by all customers.

After a short lunch break, Sandrine and I presented the new electronic b-commerce system based on BBj® that allows all customers to order new licenses or acquire needed upgrades on demand 24x7. The positive feedback of customers pointed out that BASIS has put the right focus on this client-oriented system, satisfying many customer wishes.

To give the enthusiastic partners some insight into the development roadmap of the upcoming BBj 6.0, I explained the new features of this release such as triggers, stored procedures, custom objects, VKEYED files, and the new IDE-integrated AppBuilder. The positive feedback from the partners showed that the continuous BASIS engineering and development have implemented, once more, the needed extensions and improvements that have brought BBj to the point where it is today: the most powerful and readily market-available BBx®.

Finally, Nico thanked all those present for coming and their lively and active participation in the event. He pointed out that the Partnership Program and the continuous communication between BASIS and their customers are important steps towards a common strategic market expansion.

The Partnership Program Launch Tour 2006 continued through the wintry white Germany to Düsseldorf, Frankfurt, and finally Munich, where the journey ended. During the trip, the business atmosphere among participants continuously improved due to the positive customer feedback and enthusiastic partners we encountered in every city. After traveling almost 1200 miles across Germany, our BASIS team found ourselves very confident and grateful that we could present the new Partnership Program to more than 60 satisfied customers in a fruitful dialog.

Johannes Fritz
Chief Information Officer, BSG GmbH

The FAQs About 64-Bit BASIS Products

By Janet Smith

With an increasing number of operating systems vendors offering 64-bit versions, BASIS Technical Support is often asked if BASIS offers 64-bit products that will run on these operating systems. This article answers which BASIS products are available in 64-bit versions and common questions regarding downloading and installing.

Q Does BASIS offer any 64-bit ports? How do I know if there is a 64-bit product available for my operating system?

A Yes, BASIS introduced 64-bit products for BBj® 6.0 and PRO/5® 6.0. In addition to a 64-bit operating system, this version of BBj 6.0 requires the installation of a 64-bit JVM (Java Virtual Machine) on the operating system. A complete listing of our products and operating systems is published on the Platform Availability page at www.basis.com/products/availability.html.

Q How do I read the Platform Availability report?

A The Platform Availability list is organized in sections; first by product, then by operating system. Locate the section for the product you wish to install (BBj, PRO/5, etc.) then look for your specific operating system. To the right of the operating system appears the associated BASIS port ID, revision level, and the product release date. The BBj product list also includes the Java version under which we tested BBj. Likewise, the PRO/5, Visual PRO/5®, and ODBC products show the operating system levels used for testing.

Q How does this report differentiate a 32-bit port from a 64-bit port?

A A *(64-bit)* notation appears after each operating system description that offers a 64-bit product as shown in **Figure 1**. Additionally, all 64-bit port IDs start with the number 6.

Compaq Tru64 UNIX *(64-bit)* *Java 1.4.2-5*	6175	**6.0**	05-Jun-2006

Figure 1. A port with a 64-bit notation

The Platform Availability page also indicates when a 64-bit product is bundled with a 32-bit product for BBj; both port IDs for the operating system appear in the same color band. For the PRO/5 product family, the 32- and 64-bit ports are not bundled together and exist separately in the list of available ports.

Figure 2 and **Figure 3** compares the appearance of a bundled 64-bit port and a 64-bit only port.

IBM AIX **IBM AIX** *(64-bit)* *Java 5.0.0.1 (AIX 5.3)*	2184 6184	**6.0**	05-Jun-2006

Figure 2. A 64-bit port bundled with the 32-bit version

Linux Power5 *(64-bit)* *Java 5.0*	6146	**6.0**	05-Jun-2006

Figure 3. An unbundled 64-bit port with no available 32-bit version

Q Okay, I found a 64-bit BBj port for my operating system and verified my installed JVM is the correct version. How do I download the correct product from the download page?

A First, go to "BBj Downloads" on the BASIS **Products | Downloads** Web page. Select the desired release type such as "Current Release Product Version 6.0" from the **Release Type** section. Once the release is selected, select the platform that corresponds to your Operating System from the **Platform** section. The selection should indicate that a 64-bit BBj is available for your Operating System.

Once you choose the correct operating system and complete the download form, you will receive a download file that starts with either a **2** (**3**2-bit) or **6** (**6**4-bit). If your download filename begins with a **2**, the 64-bit port is bundled with the 32-bit product. Most 64-bit ports are bundled with the 32-bit ports' download jar file. This means the same single download file will install both 32-bit and 64-bit ports on their respective operating systems.

Q If the 64-bit BBj product I downloaded is bundled with 32-bit product, how do I install the 64-bit product?

A Start the installation as normal. The screen shown in **Figure 4** appears during this process and requests the location of Java (JVM) version installed on the system. Simply select the 64-bit JVM and the install correctly configures BBj to use the 64-bit JVM.

continued...

Janet Smith
Technical Support Supervisor

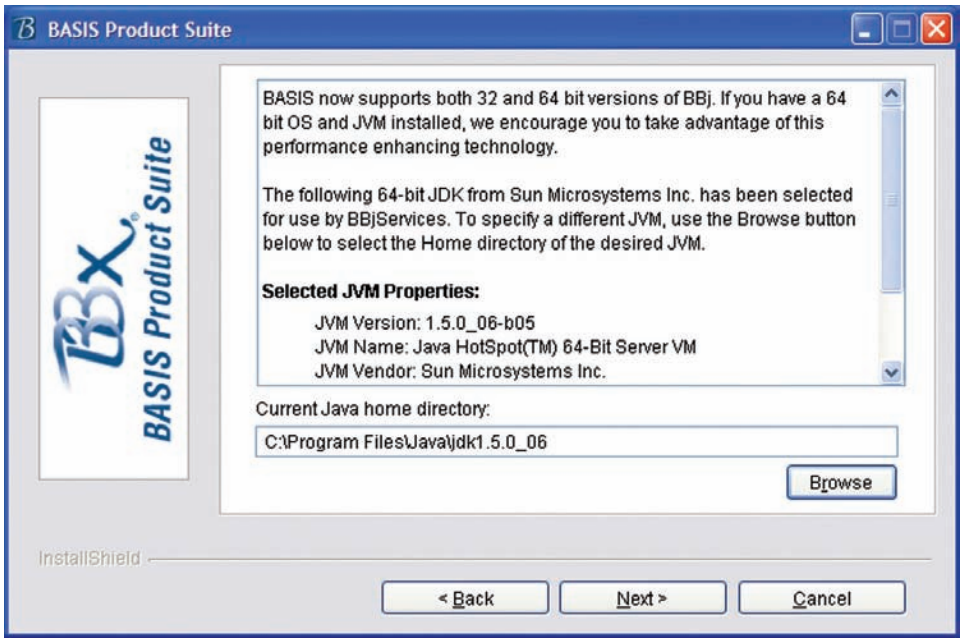


Figure 4. Screen shot requesting the location of the 64-bit JVM

For more information about this installation, refer to the
- Installation notes in the BBj 6.0 readme at www.basis.com/products/bbj/readme600.htm
- Online documentation for installing BBj
  for Windows www.basis.com/onlinedocs/documentation/whcsh_home.htm#id=27000
  for UNIX www.basis.com/onlinedocs/documentation/whcsh_home.htm#id=23003

Q I have both a 32-bit and 64-bit JVM installed on my system. When I ran the installation, I incorrectly chose the 32-bit JVM. Do I have to re-install BBj before I can use the 64-bit JVM?

A No, you can run Enterprise Manager to correctly configure BBj Services to use the 64-bit JVM. Follow these steps:

1. Connect to the server running BBj.
2. Enter the user name and password. The default admin user is admin with the default password of **`admin123`**.
3. Double click on the "JVM Settings" node. The Application Java Settings dialog box (**Figure 5**) now appears, displaying **`Default`** as the "Application" type and the pathname of the JVM that BBj is currently using, as the "Java Home" field.
4. Click [Browse…] and locate the correct location of the 64-bit JVM.
5. Click [OK].
6. Double click on the "JVM Settings" node once again.
7. Click on the drop down menu in the "Application" box and select "BBjServices."
8. Browse to same location used for the "Default" JVM.
9. Exit Enterprise Manager
10. Run the BBj Admin utility and stop and start BBj Services to apply the change.

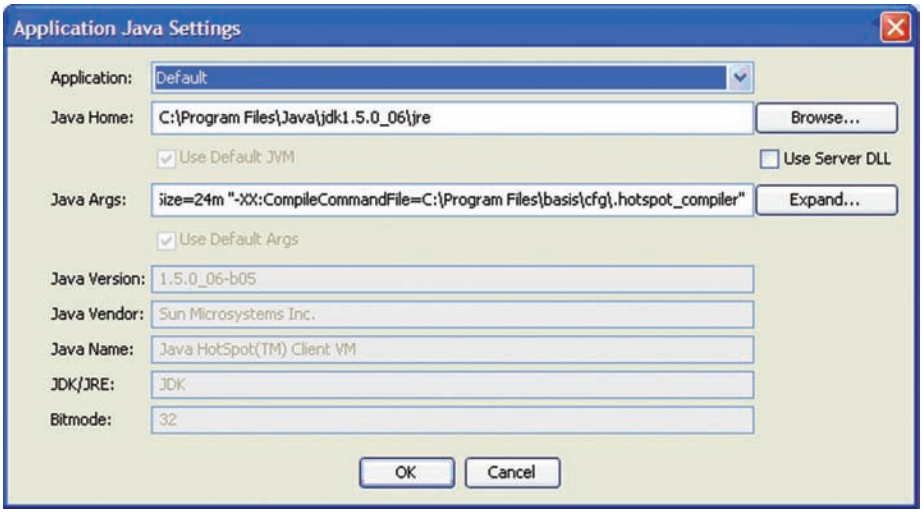


Figure 5. The Java setting dialog box in Enterprise Manager

Exercise caution when using this method. Selecting an incorrect JVM will prevent BBjServices from starting. If this situation occurs, uninstall and reinstall BBj to correct the problem.