Goodbye Event Loop, Hello Process Events

By Jon Bradley

For years, BASIS has provided tools to facilitate writing GUI applications. BASIS is at it again, improving the tools developers use to create their applications!

Background

When BASIS incorporated the functionality of GUIBuilder into the IDE, they decided to continue to read and write the same GBF and ARC files. This makes it easy to move from GUIBuilder to the new, more feature-rich AppBuilder. AppBuilder also uses the same back-end code generator, gb_func, to generate the same READ RECORD-based applications as GUIBuilder. In the preview releases of BBj® 9.0, AppBuilder exposes the generator interface to enable developers to easily modify – or even completely replace – the generator module.

Goodbye, gb_func

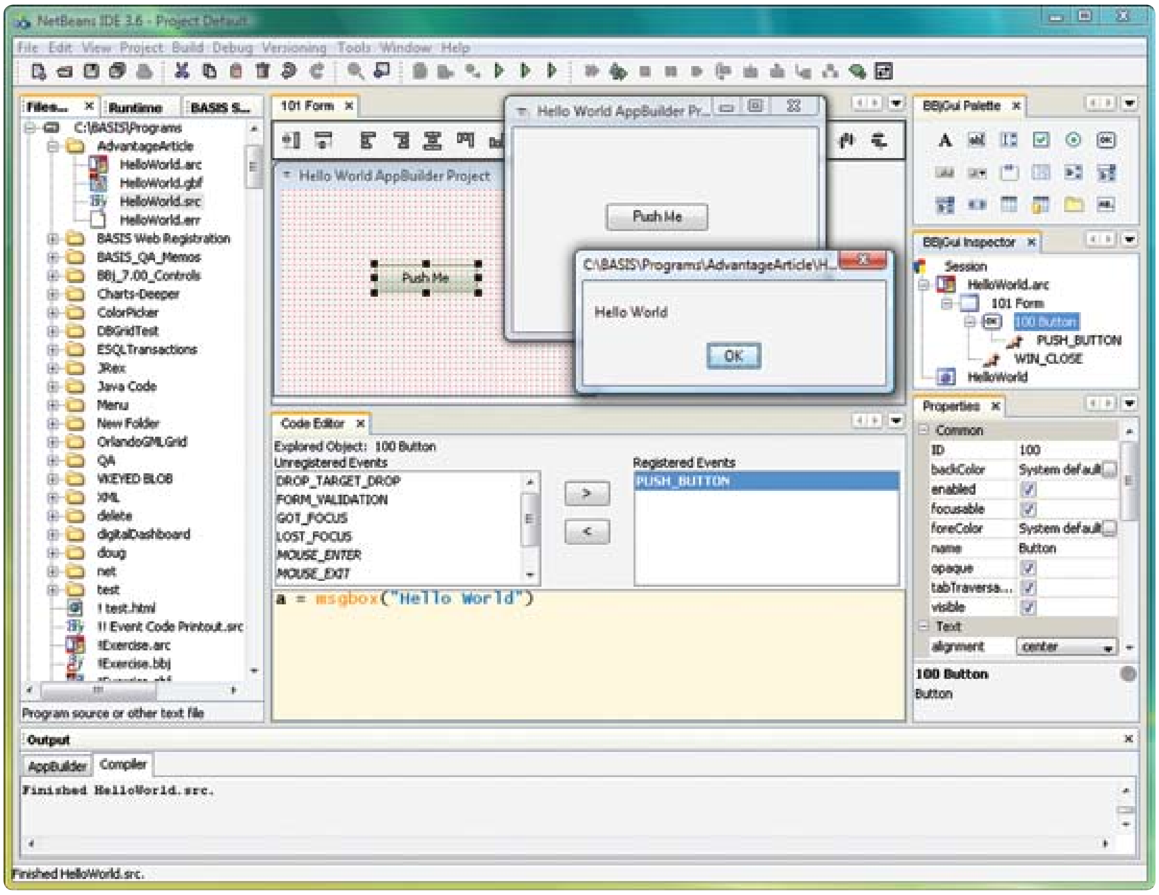
Both GUIBuilder and AppBuilder use a generator program to convert the .gbf and .arc files into a deliverable application. Prior to BBj 9.0, AppBuilder always used the same gb_func generator program as GUIBuilder, thus it produced the exact same application code. BBj has so many new language constructs ranging from PROCESS_EVENTS to custom objects that BASIS wanted to create a new generator program to exploit the new language features and generate more powerful, readable, and efficient programs.

Hello, ProcessEventsBuilder

Instead of dictating what form the newly generated application code takes, BASIS makes it easy to write a custom generator program while supplying two sample generator programs to get started. The ProcessEventsBuilder generator and its close cousin, LegacyPEBuilder, both use the PROCESS_EVENTS method for dispatching events. This method represents a more modern, succinct, and efficient way to handle events.

When using PROCESS_EVENTS, developers register callbacks for the specific events on specific controls for which they are interested. The dispatch of that event to a given routine happens automatically, reducing the amount of boilerplate

continued...


NetBeans IDE 3.6 - Project Default
File Edit View Project Build Debug Versioning Tools Window Help
Files... Runtime BASIS S...
101 Form
Hello World AppBuilder Project
Push Me
Hello World
OK
BBjGui Palette
BBjGui Inspector
Code Editor
Properties
Output
AppBuilder Compiler

Explored Object: 100 Button

Unregistered Events: DROP_TARGET_DROP, FORM_VALIDATION, GOT_FOCUS, LOST_FOCUS, MOUSE_ENTER, MOUSE_EXIT

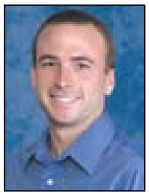
Registered Events: PUSH_BUTTON

```
a = msgbox("Hello World")
```

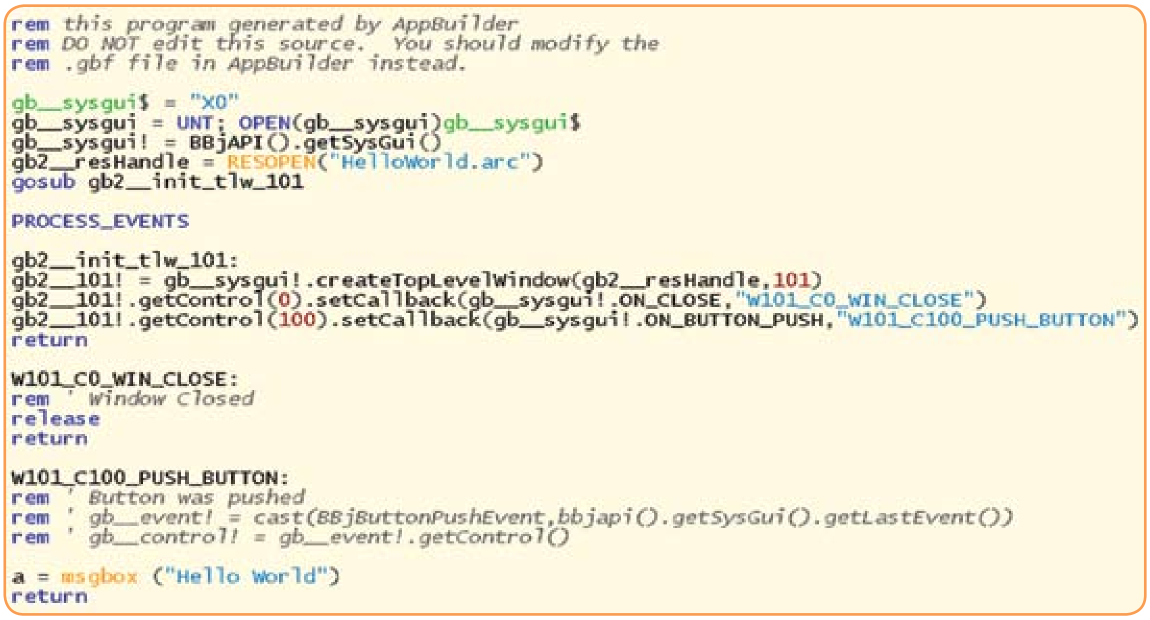
Common	
ID	100
backColor	System default
enabled	✓
focusable	✓
foreColor	System default
name	Button
opaque	✓
tabTraversa...	✓
visible	✓
Text	
alignment	center

```
Finished HelloWorld.src.
```

Figure 1. "Hello World" AppBuilder project



Jon Bradley
Software Engineer



```
rem this program generated by AppBuilder
rem DO NOT edit this source.  You should modify the
rem .gbf file in AppBuilder instead.

gb__sysgui$ = "X0"
gb__sysgui = UNT; OPEN(gb__sysgui)gb__sysgui$
gb__sysgui! = BBjAPI().getSysGui()
gb2__resHandle = RESOPEN("HelloWorld.arc")
gosub gb2__init_tlw_101

PROCESS_EVENTS

gb2__init_tlw_101:
gb2__101! = gb__sysgui!.createTopLevelWindow(gb2__resHandle,101)
gb2__101!.getControl(0).setCallback(gb__sysgui!.ON_CLOSE,"W101_C0_WIN_CLOSE")
gb2__101!.getControl(100).setCallback(gb__sysgui!.ON_BUTTON_PUSH,"W101_C100_PUSH_BUTTON")
return

W101_C0_WIN_CLOSE:
rem ' Window Closed
release
return

W101_C100_PUSH_BUTTON:
rem ' Button was pushed
rem ' gb__event! = cast(BBjButtonPushEvent,bbjapi().getSysGui().getLastEvent())
rem ' gb__control! = gb__event!.getControl()

a = msgbox ("Hello World")
return
```

Figure 2. Resulting code produced from building the AppBuilder project

code and the number of instructions required to process an event in the interpreter. The GUI sends fewer messages to the server, reducing network utilization and improving responsiveness.

The ProcessEventsBuilder generator program produces object-oriented code, resulting in a program that is easier to read and debug. It makes use of BBjEvent objects, which provide intuitive method names for exposing the information – information that previously appeared in hard-to-understand notify string flags.

Case in Point

In the simple "Hello World" AppBuilder project shown in **Figure 1**, the `ProcessEventsBuilder.src` generator program produced 24 lines of code shown in **Figure 2**.

This increases the readability and eases the debugging process because the resultant program is much smaller. The gb_func-generated program for the same AppBuilder project is a whopping 712 lines of code! This enormous difference is due to the large amount of code gb_func-generated programs needed in Visual PRO/5® to get and set information for the SYSGUI system. Generating a PROCESS_EVENTS program also allows the control validation system to work normally, without having to add the extra call to setCallback in the init block, which is necessary when creating a READ RECORD-based program.

Building a Custom Generator Program

BASIS provides the new AdvancedGBFWrapper Java object to expose the contents of the GBF file to the generator program in an object-oriented way. This makes writing new generator programs relatively painless. For instance, the ProcessEventsBuilder generator program is only 200 lines of code. The BASIS-provided generator programs use this same model and present a good springboard for those developers looking to take the plunge and create their own generator program.

By default, BBj continues to use the gb_func program generator. Select a different generator program via **Tools > Options > Form & AppBuilder Settings > Custom Generator Program** in the BASIS IDE.

Summary

Because the generator programs are customizable, it is easy to leverage the BBj language enhancements to create more readable and powerful applications. The application will benefit from a substantial reduction in the lines of code generated; in our "Hello World" example, the number of lines went down from 712 to just 24! Some developers have applied their own enhancements and modifications to gb_func to address their needs to create custom application code. The **AdvancedGBFWrapper** meets this need by facilitating the creation of a custom generator program that produces the application code. This delivers the desired structure when using AppBuilder for WYSIWYG editing of applications within the BASIS IDE.

Read more about design guidelines of a generator in *Working with Custom Program Generators*
www.basis.com/solutions/UsingCustomAppBuilderGeneratorPrograms.pdf