

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.

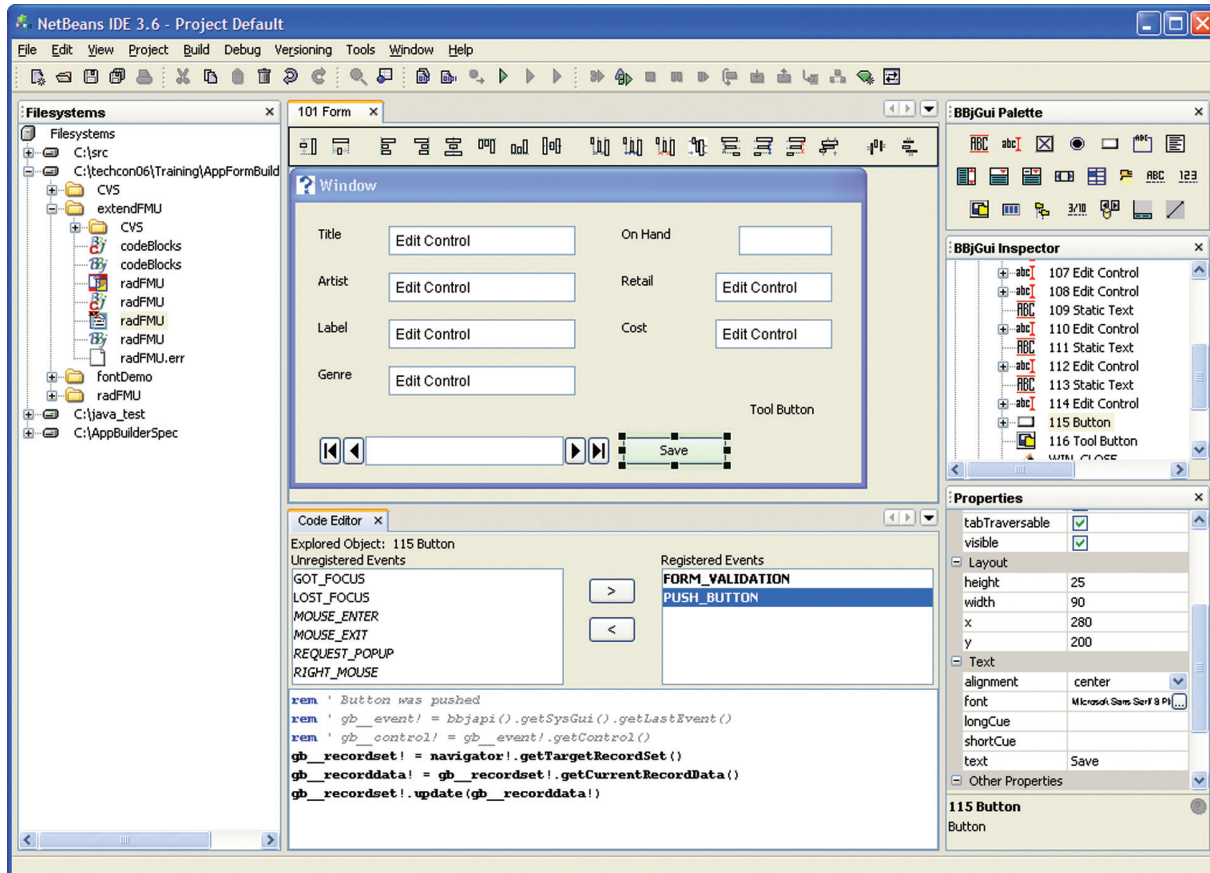


Figure 1. The AppBuilder desktop

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**.

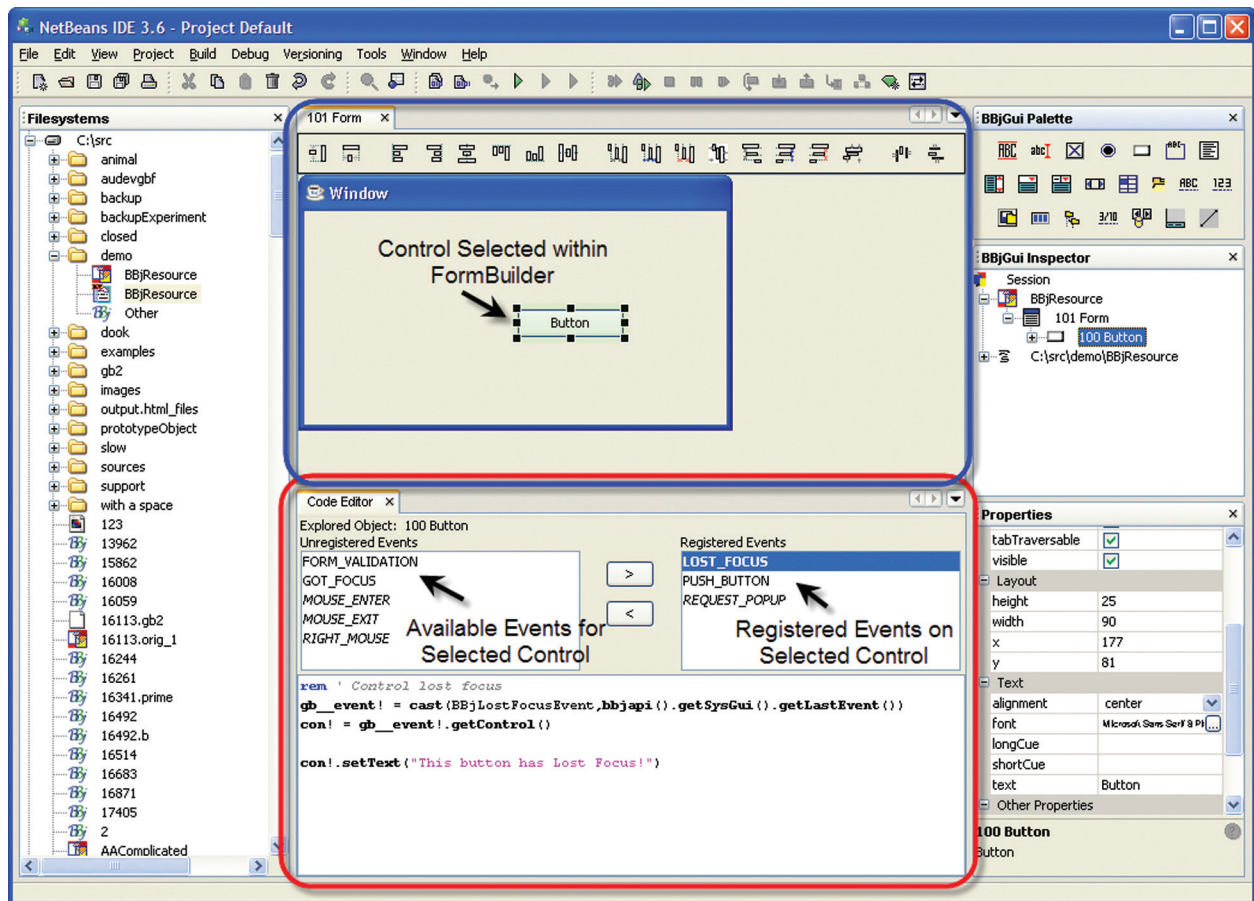


Figure 2. FormBuilder's Form Editor (blue outline) and AppBuilder's Code Editor (red outline)

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...

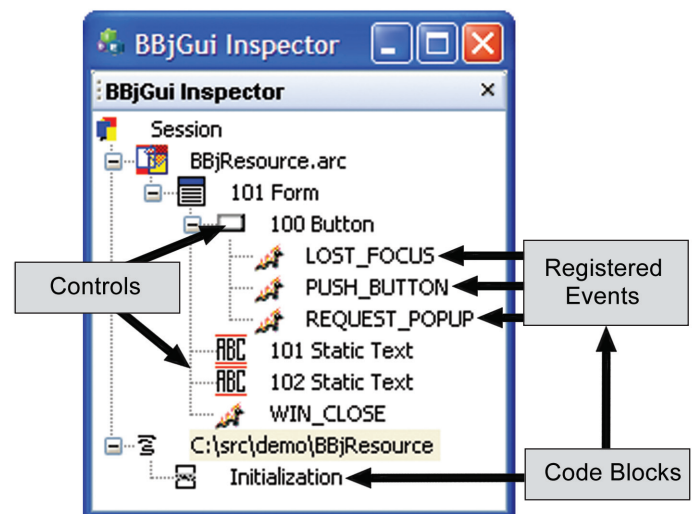


Figure 3. BBjGui Inspector

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**).

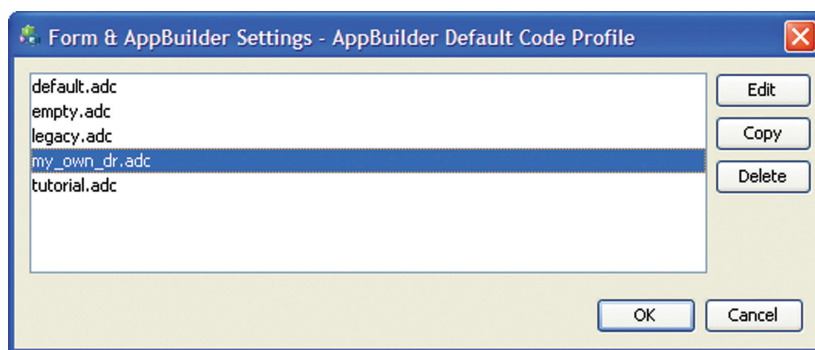


Figure 4. Default Code Profile dialog box

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**).

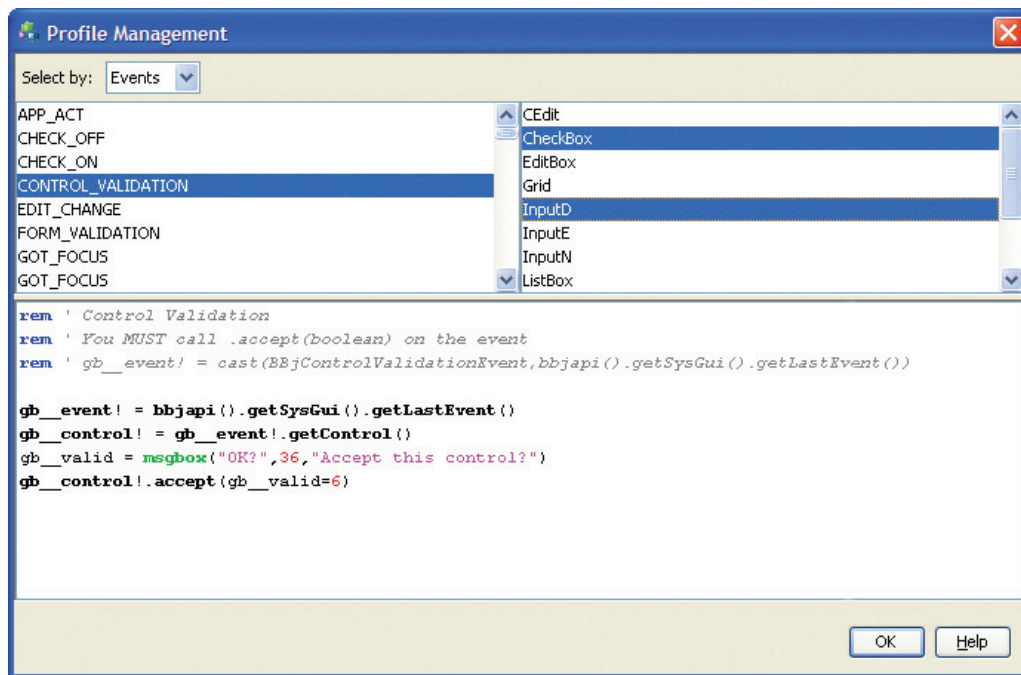


Figure 5. Profile Management dialog box

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...

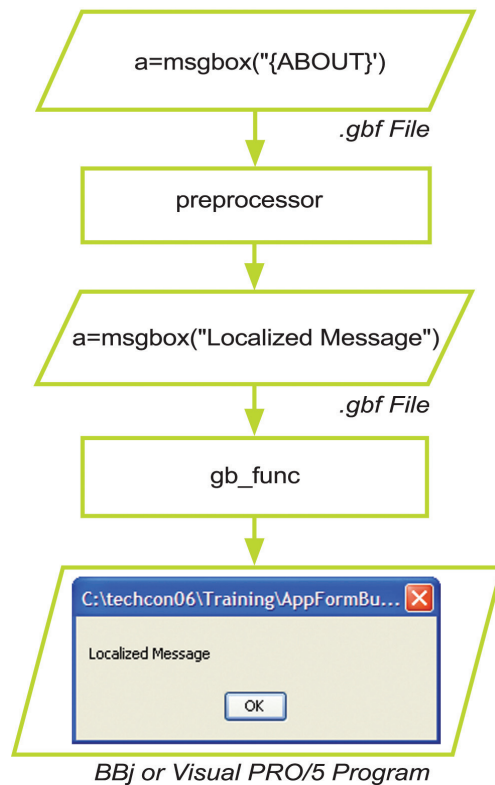



Figure 6. Sample string literal substitutions

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