

The State of the IDE

By Nick Decker and Mike Phelps

The BASIS Integrated Development Environment (IDE) arrived with several major enhancements in the distinctive 5/5/05 release of BBJ® 5.0. The most celebrated introductions and enhancements included:

- **Updated BASIS IDE** – BASIS updated the IDE’s foundation from NetBeans version 3.5 to version 3.6, offering the latest NetBeans technology with an enhanced windowing system.
- **Introduced FormBuilder** – BASIS introduced a new and robust plug-in module to replace the standalone MS Windows-only ResBuilder® utility, allowing developers to create graphical front ends for their GUI programs on any BBJ-supported platform.
- **Improved Debugger** – BASIS improved the debugger’s integration with NetBeans so it looks and operates more like Java’s debugger, making it easier to work with and easier to get things done.
- **Enhanced Source Editor and Debugger** – BASIS integrated BBJ object syntax code completion, making writing BBJ coding more convenient.

BASIS is currently developing equally ambitious enhancements for the BASIS IDE for BBJ’s 6.0 release, including a new comprehensive GUI application development tool and more convenient ways to execute code. Here is a sneak peek at a few of these IDE enhancements and those planned for the future.

Code Completion: What can this object do?

Code completion saves development time by offering a convenient type of on-the-spot documentation for BBJ Objects and quick, error-free entry of their needed parts. This feature displays a popup window that lists the variables and methods belonging to a BBJ Object whenever the code editor senses the need to complete a program statement. Simply select an item from the popup window to paste it into the edited line automatically, thus preventing syntax errors.

Since the Business BASIC language does not yet provide a standard variable type declaration syntax like Java or C/C++, the developer must declare BBJ Object variables (variable names with the ! character at the end) manually so the system can determine what to display in the popup window. Declaring a variable type simply means the developer links a BBJ Object variable name with some type of BBJ API or Java object. Upon establishing this link to an object type, the IDE Source Editor or Debugger identifies what kind of object the variable name represents and displays the correct information in the code completion popup window. In addition, it also stores variable type declarations in a special NetBeans project-specific file called **BBJObjectTypes.txt** for future use, eliminating the need to reestablish the declarations.

To see how this works, open a file in the IDE and enter a new BBJ Object variable name followed by the ! character. Notice that after entering the ! character the editor displays a popup window proclaiming “Undeclared BBJ Object” (see **Figure 1**).

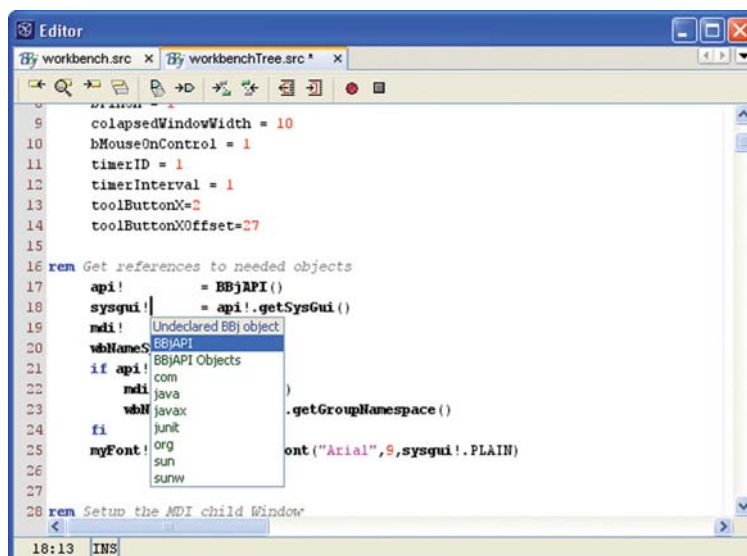


Figure 1. The variable declaration popup window

Next, browse the popup window with the mouse or the arrow keys and make selections using the [ENTER] key or by double-clicking the selection with the mouse. Selecting a category such as “BBJAPI Objects” repaints the popup window with a new listing of the objects belonging to that category. Selecting the name of a specific object completes the variable declaration process. The next time that you reference that object in the editor, the code completion system will be armed and ready.

continued...



Nick Decker
Engineering
Supervisor



Mike Phelps
Software
Engineer

The IDE also has a variable declaration editor dialog (**Figure 2**) with some helpful capabilities. Use the Edit Declarations tab of this dialog to easily add, edit, or delete declarations. The Import Declarations tab allows the importing of variable declarations already used in other files, saving time and effort.

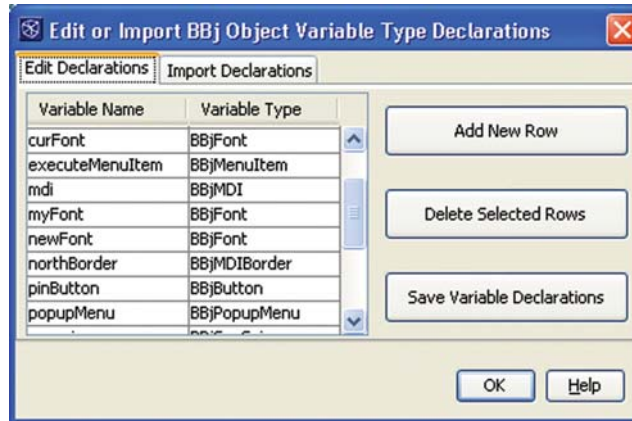


Figure 2. The variable declaration edit/import dialog

The IDE Source Editor and Debugger pay close attention to BBj Object variable names. For example, immediately after entering the . character at the end of the name of a declared object variable, the code completion system activates and displays a popup window listing every property and method belonging to that object type. **Figure 3** shows the code completion popup window for a BBjAPI type object named `api!`. Just as in the variable declaration popup window, scroll through the listings in the code completion window with the mouse or arrow keys and make a selection using the [ENTER] key or by double-clicking the selection with the mouse.

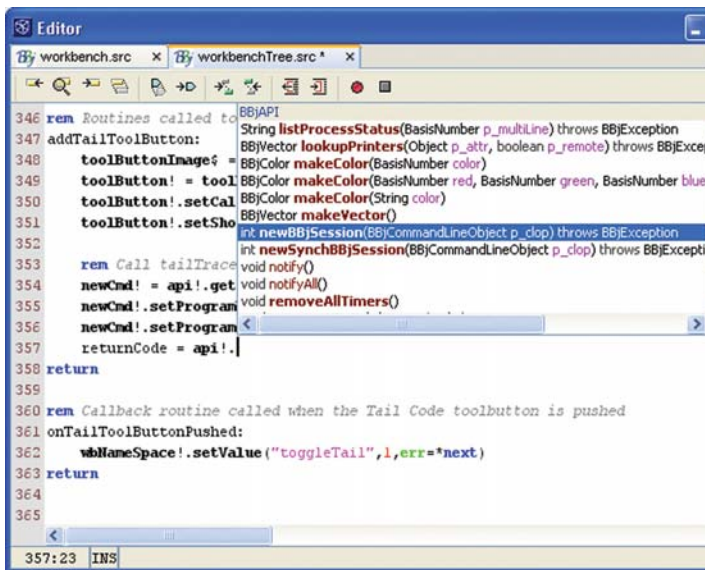


Figure 3. The code completion popup window

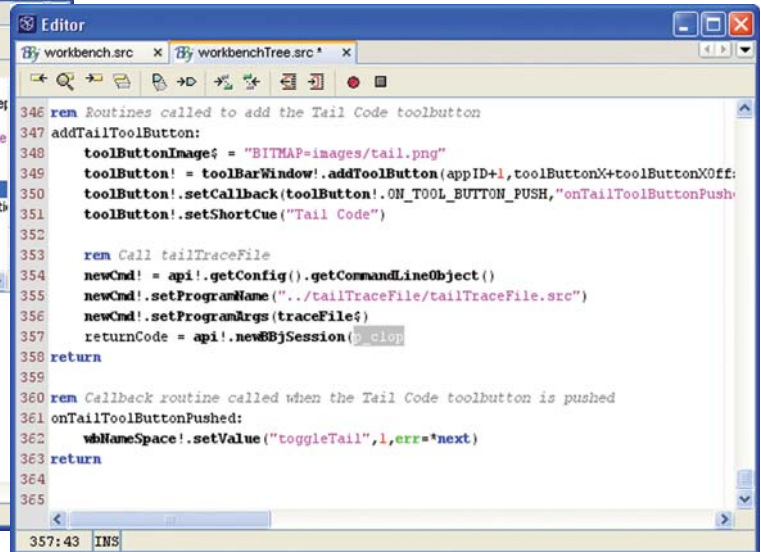


Figure 4. The code completion result

Figure 4 shows the result of selecting the method called “newBBjSession.” Selecting the method template automatically pastes the method name into the editor just after the variable name, and it prompts for the first method parameter.

Code completion is a very enticing reason for using the IDE to develop in BBj, and very difficult to live without once a developer experiences it. There is not enough space in this article to discuss code completion completely, refer to the [online BASIS IDE documentation](#) for more information.

Thin Client Execution: When all Your Code Stays on the Server

Each software development company has its own custom way of developing applications. Some organizations choose to keep and test their source code on a central server instead of on individual developer machines. The IDE makes this development methodology easier with a new feature that optionally starts a thin client connection to a remote server which executes the code. In this approach, all the application’s source files and resources remain on the server rather than being copied to each developer’s client machine. Each client machine needs only a copy of BBj and the BASIS IDE. Then simply add all the BBj execution options available from the command line to the BBj execution settings found in the Tools→Options window (see **Figure 5**).

continued...

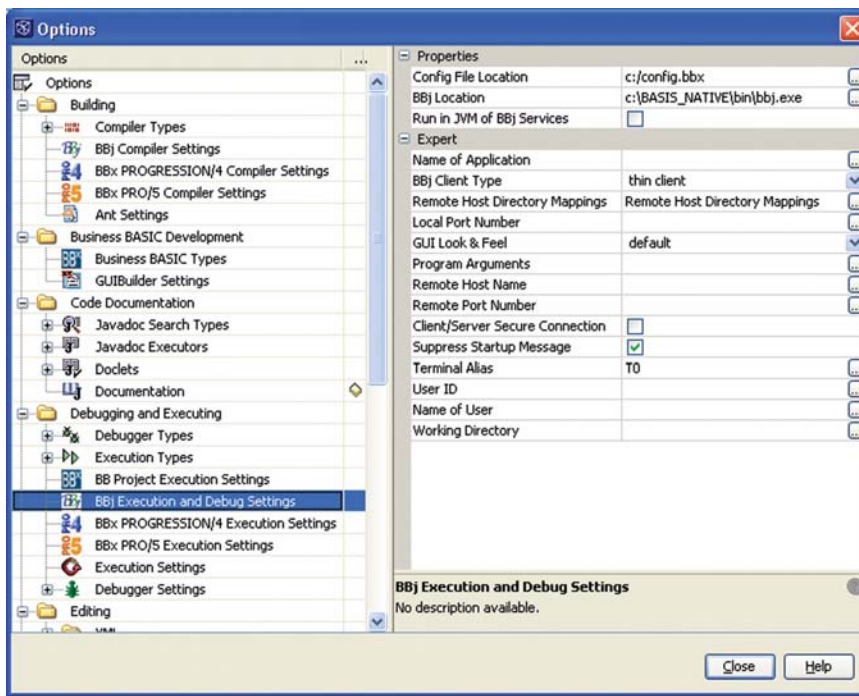


Figure 5. New BBj execution and debugging options

Critical information for completing these settings between the client and the server machines is the drive mapping definition for Windows or network mount information for UNIX/Linux, depending on the operating system. In order to mount the server's drives in the IDE Explorer, make sure that they are visible on the organization's local area network (LAN). The drive path or mount point name that appears to clients on the LAN is not, however, the same as the directory path name that the server itself uses to access the files. The BASIS IDE, running on a software developer's client machine, knows the application's files by the drive path/mount point name on the LAN, but also requires the



Figure 6. The remote host directory mappings editor dialog

corresponding server's directory path name. The server directory path name is not available to clients on the LAN, yet the BASIS IDE requires this information in order to tell BBj running on the server where to find the file to execute.

To configure the BASIS IDE for thin client execution, obtain this drive path mapping information from your LAN administrator and store it in the BBj execution settings. Clicking the button in the Remote Host Directory Mappings option opens the editor dialog shown in Figure 6.

Enter the server directory or directories mounted in the IDE Explorer under the "Client Mounted Directory" column, and the corresponding actual directory paths under the "Remote Host Directory" column. At execution time, the IDE takes the complete directory path name of the file selected for execution, substitutes the mounted portion of the name for the actual path name (which comes from the stored directory mappings), and forwards the modified path name to BBj running on the server.

The Remote Host Name option, known as `-RH` on the command line, controls the type of execution (see Figure 5). If this option is blank, execution occurs on the local machine. If the option specifies a server, the IDE creates a BBj Thin Client on the developer's client machine. The Thin Client then connects to the specified server, which executes the application.

Leaving the Config File Location (`-c` on the command line) option blank takes advantage of the default configuration file on the server. If execution requires a specific configuration file that is not the default, the developer must identify this file using the actual drive path name known to the server. An automatic path name substitution is not possible here because there are no configuration files mounted or selected in the IDE Explorer.

Legacy BBx Execution: Closing the Loop

The BASIS IDE increases the level of support for older versions of Business BASIC. Currently, it is possible to edit and compile BBxPROGRESSION/4® or PRO/5® code in the IDE, however to run that same code in the IDE requires a BBj interpreter. If there is another interpreter installed, why not use it? Why not test BBx® application code from within the IDE on the product for which it was developed?

continued...

Why not, indeed! BASIS appropriately calls it the BASIS Integrated Development Environment because of our strong commitment to making it work with all BBx generations to the maximum extent possible. **Figure 7** shows the new BBxPROGRESSION/4 and PRO/5 execution settings in the Options window in BBj's 6.0 IDE. These new settings will operate the same way as the BBj execution settings and contain every command line option that all BBx products support.

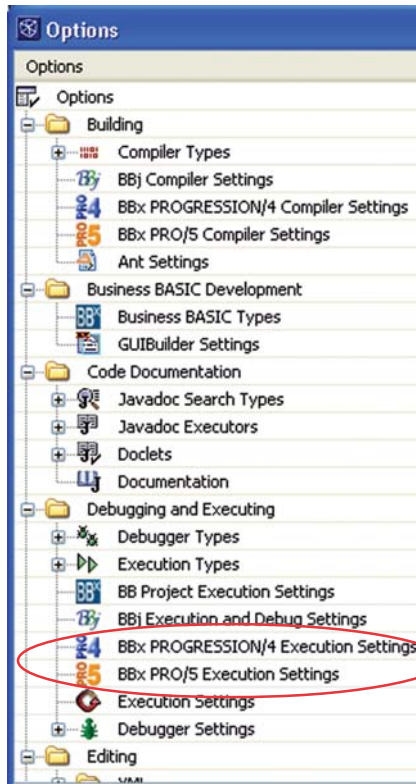


Figure 7. New BBx settings

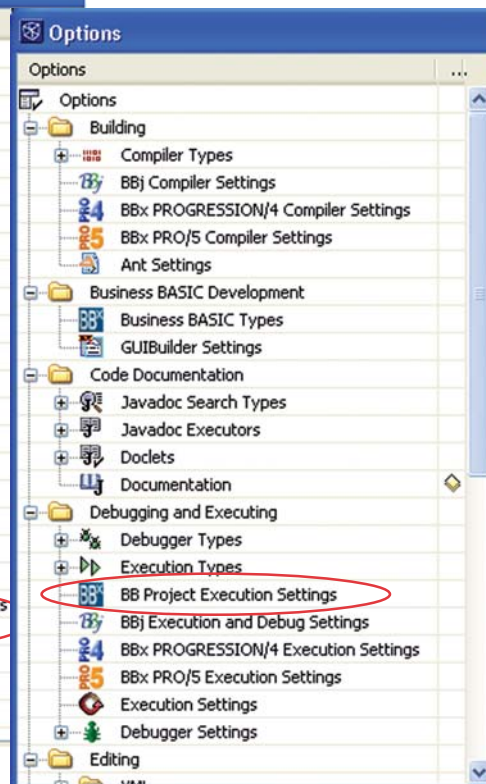


Figure 8. New Business BASIC execution settings

Project Execution: Begin at the Beginning

Another highly requested enhancement to the IDE concerns execution of an entire application instead of an individual file from within the application (see **Figure 8**). There are times when separate parts or modules from a large application require individual testing without incurring the overhead of the complete system. The IDE makes this easy - just select the specific file in the Explorer and edit, compile, or execute it. However, when it is time to integrate and test the entire application, this paradigm becomes much less convenient.

Software applications typically have a "main" program where everything begins. This program may handle user login, present the main application menu, or otherwise welcome the user to the application. It can be awkward to move back and forth between testing individual files buried deep in the application to testing the entire application. The


execution options and program arguments may need reconfiguration each time, requiring the developer to browse the Explorer for this main program file, wherever it is.

The BBj 6.0 version of the IDE allows the designation of a specified program file as the "main" or starting program of an application development project, with its own set of execution options and program arguments. When it is time to run the whole enchilada (as we say in New Mexico) rather than a single piece, select the new "Project Execute" option. The IDE executes the designated main program that launches the application at its beginning. No more wasting time reconfiguring or hunting for anything.

AppBuilder: The Integrated Graphical Application Tool

By far the most ambitious and important new feature slated for the BBj 6.0 BASIS IDE is AppBuilder, the fully integrated successor to GUIBuilder®. Regrettably, there are no pictures to preview yet. At this writing, it is impossible to discuss specifics except to say that AppBuilder will incorporate all of the capabilities of GUIBuilder integrated with FormBuilder, resulting in a single integrated graphical user interface development tool. Best of all, it will run on any BBj-supported platform.

Conclusion: Keep those cards and letters coming.

We hope you find these user-requested improvements useful and exciting. The bottom line is that your feedback and suggestions are an essential ingredient to our future planning. The BASIS IDE will continue to evolve and expand based on the needs of the Business BASIC community and your ideas, your style of work, and your vision for the future. The ultimate goal is a lofty one but it is one we are committed to – a powerful and completely integrated environment that meets all your development needs, making software development easier, faster, and more enjoyable than ever before. 



For more information about FormBuilder, refer to *FormBuilder: BASIS IDE's Better Cross-Platform Resource Builder* www.basis.com/advantage/mag-v9n1/formbuilder.html.

Refer to BASIS IDE documentation at www.basis-documentation.com/ide2/ide_features_basis.htm.