Trigger Functionality Extended with KEY Triggers

Triggers are a powerful feature of the BBj® filesystem and SQL engine, giving developers the ability to execute BBj code when certain file access operations occur such as a READ, WRITE, REMOVE, etc. BBj 11.0 introduced the ability to trigger events on calls to the KEY functions in the BBj language such as KEY(), KEYP(), KEYN(), KEYL() and KEYF().

Why is This Feature Important?

Let's take a look at a real-world example of triggers in action. The accounting group at BASIS wanted to upgrade their GL and AP modules from legacy AddonSoftware to the new 'AddonSoftware by Barista'. Though the file structures of the old and new systems were not the same, they did not want to change any code in the custom developed AR integration points to make the application read the alternate file structure. To solve this, BASIS implemented 'Instead Of' triggers to redirect operations made on the old files to the new files, specifically to redirect and read from the new files "instead" of the old files. These triggers took care of all of the application's READs and WRITEs to the data files, but calls to KEY(), KEYN(), KEYF(), KEYL(), and KEYP() functions still referenced the old data files and would not return the correct key values. Experiencing this new need, BASIS added triggers for key operations to the language and then defined the appropriate key triggers for the application's data files.

By Jeff Ash
Software Engineer

So "Why is this feature important?" The answer is simple, key triggers can handle necessary file operations and give users seamless interaction with the new files without changing the application.

Using KEY Triggers

Key triggers fire when any of the following BBj functions are called: KEY(), KEYP(), KEYN(), KEYF(), and KEYL(). Most key triggers will typically need access to two things: type of key call, and the value returned by the key call. In addition, the trigger can access things such as the key description, key number, key name, etc. For complete details, consult the BBj documentation on BBjTriggerData in the online documentation at links.basis.com/basisdocs.

The following example shows some of the calls that a typical key trigger might use:

```
declare BBjTriggerData td!
td! = BBjAPI().getFileSystem().getTriggerData()

REM Get the name of the key if it is a VKEYED file with a named key
keyName$ = td!.getKeyName()

REM Get the key number of the current key
keyNum = td!.getKeyNumber()

REM Get the key description
keyDesc$ = td!.getKeyDescription()

REM Get the key value read from disk in an after key trigger
REM NOTE: getKey() is NOT valid in a KEY trigger
keyValue$ = td!.getReadBuffer()

REM Get the type of key call that was made. Returns KEY, KEYF, KEYL,
REM KEYN, or KEYP
keyType$ = td!.getKeyCallType()

REM In an after key trigger, this will set the value returned
td!.setReadBuffer("Modified Key")
```

Figure 1. Sample Key Trigger Program

Summary

The new KEY triggers in BBj 11.0 round out the trigger functionality in the BBj filesystem. While they are not used by the SQL engine, they are common in BBj applications using READ RECORD and WRITE RECORD calls instead of SQL. With this new functionality, developers have a number of new options available when migrating users to a new version of their application. ■

Search the BASIS web site for additional trigger-related BASIS Advantage articles