New Browse Method Eases File Open Process

A common feature of web browsers is to allow the user to choose or specify a document they would like to see, and then open a corresponding application or viewer for that type of file. As websites continue to evolve towards becoming applications and vice-versa, more applications require the ability to display a document where the type of document the user will choose may not be known beforehand. This functionality, known as an application association, is already contained in popular browsers.

Wouldn't it be great if programmers could leverage the browser's functionality rather than having to roll their own? Java provides exactly this functionality through its Desktop::browse() method, and Javascript provides this functionality through Window.open(). Leveraging these two technologies, BBj® provides the BBjThinClient::browse() method in GUI and BUI.

BBjThinClient::browse() in Action

So, what exactly does this new method BBjThinClient::browse() do? Quite simply, the browse() method takes a URL as a parameter and opens that URL in a browser. URLs most commonly reference pages and files on the Web, but can also reference files locally via the `file://` protocol. Since BBj includes a Web Server, references to files on the server machine can be a URL.

To illustrate this feature, run either of the "*Browse - BUI" or the "Browse - GUI"* demos that are included with the BBj product when downloaded with the demo checkbox selected (BBj 11.10 or higher). The demo presents three options for opening a file. The first option is to specify a URL to the file, the second option is to open a file from the machine that is running BBj Services (server-side), and the third option is to open a file that resides on the same machine on which the demo is running (client-side). Once a file is specified and the "browse() file" button pressed, the file path is translated into a URL and the demo launches the default browser with the URL. The browser uses its file associations to determine the appropriate application for opening the file, and finally opens the file with that application. **Figure 1** shows the demo launching the native movie viewer application when a .mov file is passed to the browse() method.

By using the browser in this way, BBj programs can use the browse() method to open any type of file with the appropriate application without having to know which file will be specified ahead of time. For example, the default browser on a Windows system would likely open a new tab for an HTML file, launch Windows > >

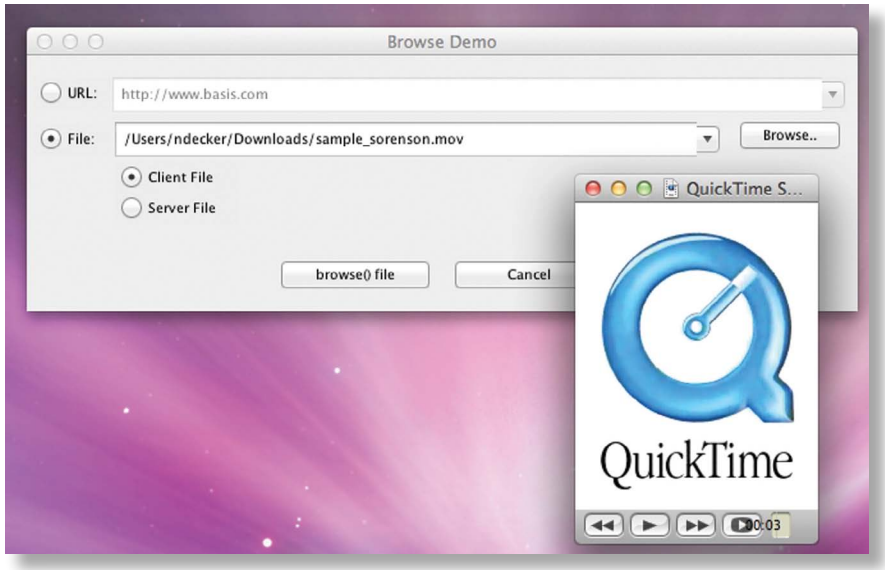

Figure 1. Browse Demo launching a native application to view the selected file

By Shaun Haney
Quality Assurance Engineer

Media Player for a .mp3 file, launch Microsoft Word for a .doc file, and launch Windows Photo Gallery for a .png file. What application opens what file, however, is ultimately in the hands of the user and how a file is opened is now outside of the realm of worry for the application developer. In many cases, the need for developers to integrate file viewers with their applications is gone.

Generally speaking, the default browser opens the specified URL and the **file://** protocol specifies URLs for local file locations. However, the browse() method treats the **file://** protocol differently from other protocols depending upon the environment in which it's used:

- The **file://** protocol is unsupported within BUI programs where the underlying Javascript mechanism does not have permissions to access the client's file system. Read on to learn how to overcome this limitation.
- Associations for URLs with the **file://** protocol may be handled by Java's URI handlers rather than being launched with a browser.

To illustrate the browse() method launching an application without the help of the default browser, enter **BBjAPI().getThinClient().browse("file:///C:/program%20files/basis/demos/adminapi/images/browse.png")** in SysConsole on a Windows Vista machine, which results in BBj opening **C:\Program Files\BASIS\demos\AdminAPI\images\browse.png** with Windows Photo Gallery. In this case, Java's URI handlers determined the file association independently of the browser on the user's system.

While the browse method delegates the task of opening files and greatly reduces or even eliminates the need for handling opening files of different types in code, a nagging question remains. Since every path passed to the browse() method must be in the form of a URL, how does one create URLs for client-side files and server-side files? The Browse demo comes in handy to answer this question since it contains code handling all of these cases.

Forming a URL for client-side files is straightforward in GUI. The URL's protocol is **file://**, followed by a beginning slash if the path does not already begin with a slash, followed by the path to the file using forward slashes as the path separator. Special characters, the most common of which is a space, require appropriate encoding for URLs. For example, if the client-side file resides at, **C:\Program Files\basis\demos\BBjThinClientBrowse\BUIBrowse.png** then the URL will be **file:///C:/Program%20Files/basis/demos/BBjThinClientBrowse/BUIBrowse.png**.

For server-side files, remember that every install of BBj comes with a Web Server (see *A Home (Page) in Every Port* at links.basis.com/11homepage) and that it runs as part of BBj Services on port 8888 by default on the server machine. Any file on the server machine placed in <BBj Install Directory>/htdocs will be accessible at **http://<server name>:8888/files/<file name>**. The Browse Demo copies the server-side file to the specified directory under the htdocs directory and then forms the URL as described above. **Figure 2** shows the result of the demo program copying the selected server file to a directory hosted by the web server and launching a new instance of the client's browser to display the file.

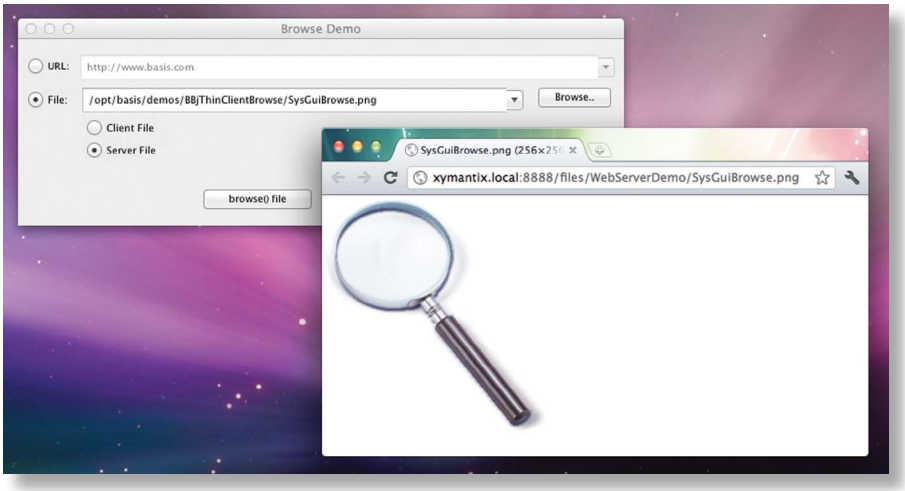

Figure 2. Browse Demo showing a server-side file in the client's browser

By way of example, if the file above resides at **C:\Program Files\basis\demos\BBjThinClientBrowse\BUIBrowse.png** on the server machine, the demo program copies it to **C:\Program Files\basis\htdocs\WebServerDemo\BUIBrowse.png**, and the corresponding URL becomes **http://<myserver>:8888/files/WebServerDemo/BUIBrowse.png**.

Opening a client-side file with the browse method in BUI is much like opening a server-side file except that the file starts out on the client-side. To view the file, it must be served up from the server-side web server. So, the first step is to create a BBjClientFile for the file that is to be browsed. Next, call the copyFromClient() method on the BBjClientFile object to copy the file to the server. The string returned by copyFromClient() reveals the location of the file on the server. Finally, copy the file over to the **htdocs** directory for the web browser to serve up.

Summary

Using the BBjThinClient::browse() method provides application developers the ability to present any BBj accessible file to the user, both client and server-side, without needing to determine ahead of time which application needs to be associated with the file for the user to view its contents. The user's machine configuration determines how the file will be opened and avoids the need for the developer to provide this knowledge from within the application. This BASIS-supplied functionality enables desktop and web application convergence, allowing developers to create applications that deliver content in a greater variety of formats regardless of the mechanism of their deployment. BASIS continues to deliver on its promise of "*Write once, run EVERYwhere*"! ■

For more information, refer to online documentation

- Desktop::browse
- BBjThinClient::browse