Tcl/Tk and the .NET Framework

written by Joe Mistachkin

Tcl/Tk integration with Microsoft RAD environments in the COM era

In the past, it took considerable effort to successfully utilize Tcl/Tk from any of the Microsoft RAD environments (Visual Basic "Classic", VBScript, Internet Explorer, JavaScript, etc).

There are various limitations in these environments that make directly accessing the Tcl C API very difficult, if not impossible.

TclBridge "Classic"

I began developing TclBridge in 1999 as a way for me to script my Visual Basic applications. Since then, it has evolved from its original design and is currently used worldwide with a wide variety of programming languages, environments, and third-party software packages.

Tcl/Tk integration with .NET

Since .NET was released in early 2002 it has grown dramatically in popularity. However, there is still no easy way to add rich dynamic scripting capabilities to .NET applications.

TclBridge.NET

In early 2004, I began development of a native .NET version of TclBridge. The original goals were that it would be written 100% in .NET and provide a superset of the functionality in TclBridge "Classic".

At this point, TclBridge.NET is still in the early stages of development. It does not yet provide a superset of the functionality of TclBridge "Classic" and it still requires a small native code (C/C++) DLL to facilitate Tcl custom command callbacks implemented as methods on .NET objects.

Tcl runtime resources

Tcl keeps track of several types of resources that host applications generally need to be concerned with.

The first resource type is Tcl interpreters (Tcl_Interp structures). Tcl interpreters exist until they are explicitly deleted or the process exits. The basic rule for .NET objects is that they exist only until they go out of scope. Typically, this will be the scope will be a function, the lifetime of a containing object, or until the process exits.

The second resource type is Tcl objects (Tcl_Obj structures). These structures may contain any of the built-in Tcl object types or types exposed by extensions.

The third resource type is Tcl strings. Technically, there are several types of strings that Tcl is capable of managing; however, in practice, host applications will deal primarily with UTF-8 or Unicode strings with known lengths contained within Tcl objects.

Client applications versus server applications

Traditionally, client applications are able to maintain "state" and server applications are not. There are several special considerations when scripting server applications.

- Is the server application able to maintain the Tcl interpreter between requests?
- Does each user that hits the server need their own Tcl interpreter?
- Can a Tcl interpreter be recycled for use by another user without compromising security? If not, are there any negative consequences from creating and deleting a Tcl interpreter for each request?
- Are all the scripts on the server trusted?
    - What level of trust?
    - What users may add new scripts or modify existing scripts?
- How are errors handled?