The wonderful world of shell bind context strings

devblogs.microsoft.com/oldnewthing/20150122-00

January 22, 2015

Raymond Chen

Some time ago, we saw how the `IBindCtx` parameter to `IShellFolder::ParseDisplay-Name` can be used to modify how the parse takes place. More generally, the `IBindCtx` parameter to a function is a catch-all *miscellaneous options* parameter.

The interesting part of the bind context is all the stuff that has been added to it via the `IBindCtx::RegisterObjectParam` method. You can attach arbitrary objects to the bind context, using a string to identify each one.

Some bind context parameters are like Boolean parameters that simply change some default behavior of the operation. For these operations, the object that is associated with the bind context string is not important; the important thing is that there is *something* associated with it. Traditionally, you just connect a dummy object that implements just `IUnknown`.

In the most general case, the object associated with a bind context string implements some operation-specific interface. For example, the `STR_BIND_DELEGATE_CREATE_OBJECT` bind context string expects you to associate an object that implements the `ICreateObject` interface, because the whole point of the `STR_BIND_DELEGATE_CREATE_OBJECT` bind context string is to say, “Hey, I want to create objects in a nonstandard way,” so you need to tell it what that nonstandard way is.

At the other extreme, you may have a chunk of data that you want to associate with the bind context string. Since bind contexts want to associate objects, you need to wrap the data inside a COM object. We saw this earlier when we had to create an object that implemented the `IFileSystemBindData` interface in order to babysit a `WIN32_FIND_DATA` structure.

Rather than having to create a separate interface for each data type (hello, `IObjectWith-FolderEnumMode`), and rather than going to the opposite extreme and just using `IStream` to pass arbitrary unstructured data, the shell folks decided to take a middle-ground approach: Use a common interface that still has a modicum of type safety, namely, `IPropertyBag`. Other nice things about this approach is that there are a lot of pre-existing helper functions for property bags and property variants. Also, you need to attach only one object instead of a whole bunch of tiny little ones.

Under this new regime (which took hold in Windows 8), the bind context has an associated property bag, and you put your data in that property bag.

In pictures:

		`IBindCtx::RegisterObjectParam`			`IPropertyBag::Write`	
`IBindCtx`	→	Boolean parameter	`IUnknown`			
		Interface parameter	object with custom interface			
		`STR_PROPERTYBAG_PARAM`	`IPropertyBag`	→	Property bag `DWORD` parameter	`VT_UI4`
					Property bag string parameter	`VT_BSTR`
					Property bag Boolean parameter	VT_BOOL

If you want a Boolean-style parameter to be `true`, then set it in the bind context with a dummy object that implements `IUnknown`. If you want a Boolean-style parameter to `false`, then omit it from the bind context entirely.

To set an interface-style parameter, set it in the bind context with an object that implements the desired interface.

To set a property bag-based parameter, set it in the property bag with the appropriate variant type.

Here are the bind context strings defined up through Windows 8.1 and the way you set them into the bind context.

Bind context string	**Model**	**Operation**
`STR_AVOID_DRIVE_RESTRICTION_POLICY`	Boolean	Binding
`STR_BIND_DELEGATE_CREATE_OBJECT`	Interface `ICreateObject`	Binding

`STR_BIND_FOLDER_ENUM_MODE`	Interface `IObjectWith-FolderEnumMode`	Parsing
`STR_BIND_FOLDERS_READ_ONLY`	Boolean	Parsing
`STR_BIND_FORCE_FOLDER_SHORTCUT_RESOLVE`	Boolean	Binding
`STR_DONT_PARSE_RELATIVE`	Boolean	Parsing
`STR_DONT_RESOLVE_LINK`	Boolean	Binding
`STR_ENUM_ITEMS_FLAGS`	Property bag: `VT_UI4`	Binding for enumeration
`STR_FILE_SYS_FIND_DATA`	Interface `IFileSys-BindData` or `IFileSys-BindData2`	Parsing
`STR_FILE_SYS_BIND_DATA_WIN7_FORMAT`	Boolean	Parsing
`STR_GET_ASYNC_HANDLER`	Boolean	GetUIObjectOf
`STR_GPS_BESTEFFORT`	Boolean	Binding for `IProperty-Store`
`STR_GPS_DELAYCREATION`	Boolean	Binding for `IProperty-Store`
`STR_GPS_FASTPROPERTIESONLY`	Boolean	Binding for `IProperty-Store`
`STR_GPS_HANDLERPROPERTIESONLY`	Boolean	Binding for `IProperty-Store`
`STR_GPS_NO_OPLOCK`	Boolean	Binding for `IProperty-Store`
`STR_GPS_OPENSLOWITEM`	Boolean	Binding for `IProperty-Store`
`STR_IFILTER_FORCE_TEXT_FILTER_FALLBACK`	Boolean	Binding for `IFilter`
`STR_IFILTER_LOAD_DEFINED_FILTER`	Boolean	Binding for `IFilter`

`STR_INTERNAL_NAVIGATE`	Boolean	Loading history
`STR_INTERNET-FOLDER_PARSE_ONLY_URLMON_BINDABLE`	Boolean	Parsing
`STR_ITEM_CACHE_CONTEXT`	Interface `IBindCtx`	Parsing and initiailzing
`STR_NO_VALIDATE_FILENAME_CHARS`	Boolean	Parsing
`STR_PARSE_ALLOW_INTERNET_SHELL_FOLDERS`	Boolean	Parsing
`STR_PARSE_AND_CREATE_ITEM`	Interface `IParse-AndCreateItem`	Parsing
`STR_PARSE_DONT_REQUIRE_VALIDATED_URLS`	Boolean	Parsing
`STR_PARSE_EXPLICIT_ASSOCIATION_SUCCESSFUL`	Property bag: `VT_BOOL`	Parsing
`STR_PARSE_PARTIAL_IDLIST`	Interface `IShell-Item`	Parsing
`STR_PARSE_PREFER_FOLDER_BROWSING`	Boolean	Parsing
`STR_PARSE_PREFER_WEB_BROWSING`	Boolean	Parsing
`STR_PARSE_PROPERTYSTORE`	Interface `IPropertyBag`	Parsing
`STR_PARSE_SHELL_PROTOCOL_TO_FILE_OBJECTS`	Boolean	Parsing
`STR_PARSE_SHOW_NET_DIAGNOSTICS_UI`	Boolean	Parsing
`STR_PARSE_SKIP_NET_CACHE`	Boolean	Parsing
`STR_PARSE_TRANSLATE_ALIASES`	Boolean	Parsing
`STR_PARSE_WITH_EXPLICIT_ASSOCAPP`	Property bag: `VT_BSTR`	Parsing
`STR_PARSE_WITH_EXPLICIT_PROGID`	Property bag: `VT_BSTR`	Parsing
`STR_PARSE_WITH_PROPERTIES`	Interface `IPropertyStore`	Parsing
`STR_PROPERTYBAG_PARAM`	Interface `IPropertyBag`	holds property bag parameters

`STR_SKIP_BINDING_CLSID`	Interface `IPersist`	Parsing and binding

There are some oddities in the above table.

- All of the `STR_GPS_*` values would be more conveniently expressed as a single `VT_UI4` property bag-based value. (Exercise: Why isn't it?)
- The `STR_ITEM_CACHE_CONTEXT` bind context parameter is itself another bind context! The idea here is that you, the caller, are enabling caching during the parse, and the inner bind context acts as the cache.
- The `STR_PARSE_EXPLICIT_ASSOCIATION_SUCCESSFUL` value is unusual in that it is something set by the parser and passed back to the caller.
- As we have been discussing, `STR_PROPERTYBAG_PARAM` is a bind context string that doesn't mean anything on its own. Rather, it provides a property bag into which more parameters can be stored.

Next time, I'll write some helper functions to make all this slightly more manageable.

Raymond Chen

Follow