Why do new controls tend to use COM instead of window messages?

devblogs.microsoft.com/oldnewthing/20090831-00

August 31, 2009

Raymond Chen

Commenter David wonders why new controls tend to use COM instead of window messages. "It seems that there must have been a decision to only develop COM controls after the invention of COM."

There have been plenty of Win32 controls invented after the invention of COM. In fact, the entire common controls library was developed after the invention of COM. All your old friends like the list view, tree view, and property sheets are good old Win32 controls. But it's true that the newer stuff tends to use COM. Why is that?

I am not aware of any grand pronouncement on this subject. Each team makes a decision that they feel is best for their customers. But if you think about it, it's not an unreasonable choice: Suppose you were writing a new C++ object. Would you prefer to use this:

```
class Thing {
public:
  enum MESSAGENUMBER {
  MESSAGE_INSERTITEM,
  MESSAGE_DELETEITEM,
  MESSAGE_DELETEALLITEMS,
  MESSAGE_SETLABELTEXT,
  MESSAGE_GETNEXTITEM,
  MESSAGE_ADDBITMAP,
  ...
  };
  LRESULT Message(MESSAGENUMBER m, WPARAM wParam, lPARAM lParam);
private:
  ...
};
```

or would you rather use this:

```
class Thing {
public:
  BOOL InsertItem(Item *item, Item *itemParent, Item *itemInsertAfter);
  BOOL DeleteItem(Item *item);
  BOOL DeleteAllItems();
  BOOL SetLabelText(Item *item, PCWSTR pszText);
  Item *GetNextItem(Item *item);
  BOOL AddBitmap(HINSTANCE hinst, PCWSTR pszResource, COLORREF crTransparent);
  ...
private:
  ...
};
```

It's just less of a hassle using separate member functions, where you don't have to try to pack all your parameters into two parameters (cryptically named `WPARAM` and `LPARAM`) on the sending side, and then unpack the parameters on the window procedure side.

The overhead of sending a message can add up for high-traffic messages. A C++ method call is pretty direct: You set up the parameters and call the method. Whereas when you send a window message, it bounces around inside the window manager until it magically pops out the other side.

Again, these are my personal remarks and are not the official position of Microsoft on anything. But if you were writing a control, which would you prefer to have to implement? And if you were using a control, which interface would you rather use?

(That said, I can't think of many common controls that are COM-based. All the ones I know about still use boring window messages.)

Raymond Chen

Follow