Adding state to the update notification pattern, part 1

devblogs.microsoft.com/oldnewthing/20240417-00

April 17, 2024

Raymond Chen

Some time ago, we looked at the update notification pattern, but in those cases, the notification carried no state.

Consider the case where you want to call a notification handler, and the handler also receives a copy of data derived from the most recent state, rather than just being called to be told that something changed and forcing them to figure out what changed.

For example, suppose you want to add autocomplete to an edit control, but calculating the autocomplete results is potentially slow, so you want to do it in the background. But while you are calculating the autocomplete results, the user might type into the edit control, and you want the final autocomplete results to reflect the most recent edit in the edit control, rather than any results from what the edit control used to contain at some point in the past.

For concreteness, I'll use C++/WinRT, but the principle applies generally. Here's our starting point:

```
winrt::fire_and_forget
EditControl::TextChanged(std::string text)
{
    auto lifetime = get_strong();

    co_await winrt::resume_background();

    std::vector<std::string> matches;
    for (auto&& candidate : FindCandidates(text)) {
        if (candidate.Verify()) {
            matches.push_back(candidate.Text());
        }
    }

    co_await winrt::resume_foreground(Dispatcher());

    SetAutocomplete(matches);
}
```

We hop to a background thread do calculate the autocomplete results, and then return to the UI thread to report the results.

There is a race condition here if a newer autocomplete result completes more quickly than an older one. For example, if the user types "`skyp`", it may take some time to find ten things that start with "skyp". But if the user then selects all the text and replaces it by typing "`b`", it probably won't take long to find ten things that start with "b". You then have this problem:

UI thread	Background thread 1	Background thread 2
TextChanged("skyp") Continue on background thread		
	FindCandidates("skyp")	
TextChanged("b") Continue on background thread	⋮ ⋮	
	⋮ ⋮ Build `matches` ⋮	FindCandidates("b") Build `matches` matches = { "bob", "boy" } Continue on UI thread
SetAutocomplete({ "bob", "boy" });	⋮ matches = { "skype" }); Continue on UI thread	
SetAutocomplete({ "skype" });		

Even though the most recent update to the edit control was to type "b", the autocomplete window shows the results back from when the text was "skyp".

We'll spend the next few days trying to fix this problem.