Displaying the dictionary, part 2: Using text callbacks

devblogs.microsoft.com/oldnewthing/20050614-18

June 14, 2005

Raymond Chen

As we noted last time, adding items to the listview takes an absurd amount of time. Today, we'll make a failed attempt at improving this because it lets me illustrate a listview technique and it lays the groundwork for the real fix next time.

Instead of creating the items in their entirety, let's set their text to `LPSTR_TEXTCALLBACK`. This is a placeholder value which indicates "I'm not going to tell you what the string is. If you need it, call me back."

```
class RootWindow : public Window
{
 ...
 LRESULT OnCreate();
 LRESULT OnNotify(NMHDR* pnm);
 void OnGetDispInfo(NMLVDISPINFO* pnmv);
 ...
};
LRESULT RootWindow::OnCreate()
{
 ...
 // item.pszText = const_cast<LPWSTR>(de.m_pszTrad);
 item.pszText = LPSTR_TEXTCALLBACK;
 ...
   // item.pszText = const_cast<LPWSTR>(de.m_pszPinyin);
   item.pszText = LPSTR_TEXTCALLBACK;
 ...
   // item.pszText = const_cast<LPWSTR>(de.m_pszEnglish);
   item.pszText = LPSTR_TEXTCALLBACK;
 ...
}
LRESULT RootWindow::OnNotify(NMHDR *pnm)
{
 switch (pnm->code) {
 case LVN_GETDISPINFO:
  OnGetDispInfo(CONTAINING_RECORD(pnm, NMLVDISPINFO, hdr));
  break;
 }
 return 0;
}
void RootWindow::OnGetDispInfo(NMLVDISPINFO* pnmv)
{
 if (pnmv->item.iItem < 0 || // typo fixed 11am
     pnmv->item.iItem >= Length()) {
  return;         // requesting invalid item
 }
 if (pnmv->item.mask & LVIF_TEXT) {
  const DictionaryEntry& de = Item(pnmv->item.iItem);
  LPCWSTR pszResult = L"";
  switch (pnmv->item.iSubItem) {
   case COL_TRAD:    pszResult = de.m_pszTrad;    break;
   case COL_PINYIN:  pszResult = de.m_pszPinyin;  break;
   case COL_ENGLISH: pszResult = de.m_pszEnglish; break;
  }
  pnmv->item.pszText = const_cast<LPWSTR>(pszResult);
 }
 if (pnmv->item.mask & LVIF_IMAGE) {
  pnmv->item.iImage = -1;
 }
 if (pnmv->item.mask & LVIF_STATE) {
     pnmv->item.state = 0;
 }
```

```
}
LRESULT RootWindow::HandleMessage(
                          UINT uMsg, WPARAM wParam, LPARAM lParam)
{
 ...
  case WM_NOTIFY:
   return OnNotify(reinterpret_cast<NMHDR*>(lParam));
 ...
}
```

Instead of setting the strings when we create the listview items, we set their texts to `LPSTR_TEXTCALLBACK`. When the listview needs the text, it sends us a `LVN_GETDISPINFO` notification, which we handle by returning the data that the listview requested.

Sidebar: In our case, obtaining the missing data is very fast. If it were slow, we could have optimized the function further by adding the line

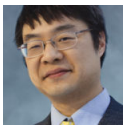
```
pnmv->item.mask |= LVIF_DI_SETITEM;
```

to the end. This tells the listview, "Please cache these results and don't ask me for them again." That way, we do the slow computation only once.

After making these changes (though not the `LVIF_DI_SETITEM` change; that was just a sidebar), notice that the it didn't really help much. On my machine, the startup time dropped from eleven to ten seconds, but ten seconds is still way too long. This optimization turns out to have been a washout.

(Note also that our program is now relying heavily on the fact that a vector is a fast random-access data structure.)

We'll do better next time.



Raymond Chen

Follow