Just change that 15 to a 1

devblogs.microsoft.com/oldnewthing/20060905-27

September 5, 2006

Raymond Chen

It would be nice and easy to just change that 15 to a 1.

If only it were that simple.

In the case described in that article, it's not that a single operation was attempted fifteen times in a loop. Rather, the fifteen operations were scattered all over the program. Suppose, for example, that the network operation was "Get the attributes of this file." The program might be filling in a file listing with several columns, one for the icon, another for the file name, another for the file author, and the last one for the last-modified time.

```
for each filename in directory {
 list.Add(new ListElement(filename));
}
```

Well, that doesn't access the same file fifteen times. Oh wait, there's more. What happens when it comes time to draw that list element?

```
ListElement::DrawIcon()
{
 if (m_whichIcon == don't know)
 {
  m_whichIcon = GetIcon(m_filename);
 }
 draw the icon for the element
}
// with this common helper function
GetIcon(filename)
{
 if (filename is a directory) {
  return FolderIcon;
 } else {
  return PieceOfPaper;
 }
}
```

Okay, getting the icon accesses the file once. You can imagine a similar exercise for getting the file's last-modified time. What else?

```
ListElement::GetAuthor()
{
 if (m_author == don't know) {
  AuthorProvider = LoadAuthorProvider();
  m_author = AuthorProvider->GetFileAuthor(m_filename);
 }
 return m_author;
}
// where the author provider is implemented in a
// separate component
GetFileAuthor(filename)
{
 if (filename is offline) {
  return "";
 } else {
  ... open the file and get the author ...
 }
}
```

Getting the author accesses the file once to see if it is offline, then again to get the actual author (if the file is online).

So in this simple sketch, we accessed the file a total of five times. It's not like there's a 5 in this program you can change to a 1. Rather, it's a bunch of 1's spread all over the place. (And one of the 1's is in a separate component, the hypothetical Author Provider.)

It reminds of a story I may have read in John Gall's *Systemantics*: There was a server product that was having problems under load. Once there were more than thirty simultaneous users, the system slowed to a crawl, but the customer needed to support fifty users. At a meeting convened to discuss this problem, an engineer joked, "Well, we just have to search the source code for the `#define` that says thirty and change it to fifty."

All the people at the meeting laughed, except one, who earnestly asked, "Yeah, so why don't we do that?"

Raymond Chen

Follow