How can I get my FileSavePicker to open in the same folder that was picked by the FileOpenPicker or FolderPicker?

devblogs.microsoft.com/oldnewthing/20220525-00

May 25, 2022

Raymond Chen

Say you want your `FileSavePicker` to default to the same folder that was picked by the `FolderPicker`. Or your `FileOpenPicker` to default to the same directory that was used by the `FileSavePicker`. Or any combination of the above. Basically, you want all the pickers to resume where the previous one left off. How do you do that?

By default, each picker keeps a separate history of recent locations, but you can override this by setting an explicit `SettingsIdentifier` on the picker. We saw this earlier when we explored how to keep two different sets of history, so that you could have a most recent location for, say, movie clips, and a different most recent location for background music.

But in addition to keeping settings separate, you can use the `SettingsIdentifier` to make them the same.

If you give your `FileSavePicker`, `FileOpenPicker`, and `FolderPicker` the same `SettingsIdentifier`, then they will share the same history of recent locations. Each one will resume in the location that the previous one left off.

```
async Task<StorageFile> LoadAsync()
{
  var picker = new FileOpenPicker {
    SuggestedStartLocation = PickerLocationId.DocumentsLibrary,
    FileTypeFilter = { ".txt" },
    SettingsIdentifier = "Common"
  };
  return await picker.PickSingleFileAsync();
}

async Task<StorageFile> SaveAsync()
{
  var picker = new FileSavePicker {
    SuggestedStartLocation = PickerLocationId.DocumentsLibrary,
    FileTypeChoices = { ["Plain Text"] = new[] { ".txt" } },
    SuggestedFileName = "New Document",
    SettingsIdentifier = "Common"
  };
  return await picker.PickSaveFileAsync();
}

async Task<StorageFolder> PickFolderAsync()
{
  var picker = new FolderPicker {
    SuggestedStartLocation = PickerLocationId.VideosLibrary,
    FileTypeFilter = { ".txt" },
    SettingsIdentifier = "Common"
  };
  return await picker.PickSingleFolderAsync();
}
```

You can combine this with the previous trick of keeping different pickers separate: You can have a group of pickers that all share their settings for movie clips, and another group of pickers that share their settings for background music.

Bonus chatter: The Win32 equivalent of the Windows Runtime `SettingsIdentifier` is `IFileDialog::SetClientGuid`. Call `IFileDialog::ClearClientData` to clean up the saved information.

Raymond Chen

Follow