

## 前提

### 技術構成のまとめ

- MemoTree は **.NET 8 / WPF の Windows 専用デスクトップアプリ**です。csproj のターゲットは net8.0-windows10.0.19041.0 で、おおよそ Windows 10 2004(20H1)以降が対象になります。
- 特徴的なのは、メモ欄を **WebView2 上の Quill 2.0.2(リッチテキストエディタ)** で実装している点です。WPF 本体とは別に、HTML(Assets/quill-editor.html)+ JavaScript でエディタを動かし、WPF⇄JS 間をメッセージでやり取りしています。
- Quill はローカルに同梱されています。従ってオフライン PC でも操作することができます。
- データはローカル保存で、%AppData%\MemoTree 配下に文書ごとの memo\_data{n}.json、5 世代のバックアップ、attachments/pdfs フォルダ、settings.json、WebView2 のユーザーデータが置かれます。
- クラウド連携やアカウントは無く、完全にローカル完結です。多重起動は Mutex で防止しています。
- NuGet 依存は 2 つだけで、Microsoft.Web.WebView2(1.0.2792.45)と Microsoft.Xaml.Behaviors.Wpf(1.1.77)です。

### ※以上のことから

- このアプリは **WebView2 ランタイム**が必須です。Windows 11 には標準で入っていますが、**Windows 10 では入っていない環境がある**ため、その場合には起動時に案内のダイアログが表示されます。画面の指示に従ってインストールをお願いします。
- また、このアプリは**.NET 8 デスクトップランタイム**も必須です。入っていない場合は、WebView2 の時と同様に起動時に案内のダイアログが表示されますので画面の指示に従ってインストールをお願いします。
- MemoTree のインストールはダウンロードした zip ファイルを解凍して、任意の場所にセットするだけです。データやセッティング情報のデフォルトの保存先は上述した%AppData%\MemoTree 配下です。  
セッティング情報の保存先は変更できませんが、データは任意の場所に変更することが可能です。

## MemoTree の使い方・仕様

- exe からの起動でタスクトレイにアイコンセット  
スタートアップに手動で MemoTree ショートカットをセットすることで、  
(Win+R→shell:startup→ショートカットセット)  
PC 起動時にタスクトレイに MemoTree アイコンがセットされる。  
タスクトレイのアイコンクリックで MemoTree の画面開く  
MemoTree の画面終了×でアイコンに戻る  
タスクトレイアイコンは常駐  
タスクトレイアイコンを右クリック→終了で MemoTree アプリが終了(Alt+F4)
- tree 型のメモ管理
- 3 文書の tree 管理
- 下階層(子)同階層(兄弟)メモ追加  
子階層追加で親階層はフォルダアイコンに変わる
- カーソル位置に画像保存
- PDF をノードに追加、画像イメージで内容表示  
データ保存フォルダに元 PDF のコピーを保存  
PDF 表示中はメモ欄は読み取り専用
- メモ単位のロック(保護)  
ロック中はエディタを読み取り専用、背景色も変更  
ツリーでの名前変更・削除もブロック  
ロックアイコンはツリー上のノード名の横に表示



文書間移動処理も不可

- ・▲▼▶◀で tree 階層のメモ移動
- ・展開折り畳みボタン
- ・ノード一覧の表示非表示ボタン
- ・常に最前面ボタン
- ・Quill ツールバーの非表示トグル
- ・あいまい検索、ノード内・文書内・すべての文書内の切替

検索結果アイテム選択でメモ表示

- ・ブログコピペボタン・・・wordpress などのブログページへのコピペで enter(段落),shift+enter(改行)を空白行挿入(段落)、通常改行(改行)でシミュレート

#### 【設定メニュー】

- ・フォントの変更
- ・文字サイズの変更・・・14px,16px,18px
- ・ライト・ダーク切替
- ・自動保存・・・初期値 60 秒 settings.json に保存
- ・文書名の変更
- ・印刷
- ・保存先の変更(デフォルトはユーザーの AppData/Roaming/MemoTree)  
データファイルのみ保存場所変更可能
- ・起動時に自動でバックアップ、5 世代のバックアップから復元可能



その他

- ・文書の保存(Ctrl+S)

- ・ノードの右クリックメニューから他文書への移動処理
- ・ホームページの構造をできるだけ維持(見た目のみ)してコピペ
- ・ネット接続オフ利用可

=====

Quill のツールバーボタンを左から

- ① Normal ▼ 見出しレベルの選択。Normal／H1／H2／H3 などを切り替える。
- ② B I U S 太字／斜体／下線／取り消し線。テキストを選択してクリック。
- ③ A \_\_ (色) 文字色／背景色(ハイライト)の変更。▼で色を選択。
- ④ = ≡ (リスト) 番号付きリスト／箇条書きリスト。
- ⑤ → ⇐ (インデント) インデントを増やす／減らす。リストの階層化にも使う。
- ⑥ 99 (引用) blockquote (引用ブロック)。左に縦線が入ったスタイルになる。
- ⑦ <> (コードブロック) 等幅フォントのコードブロック。プログラムコードの貼り付けに。  
**【重要】**コードブロックを追加するとすぐ上に「cb:」がセットされる。「cb:」は変更不可。
- ⑧  (リンク) 選択テキストに URL リンクを設定する。
- ⑨  (画像) 画像を URL で挿入。ファイルから挿入する場合はツールバーの「画像追加」ボタンを使う。
- ⑩ ≡ (配置) 左揃え／中央揃え／右揃え。
- ⑪ Tx (書式クリア) 選択範囲の書式をすべて除去してプレーンテキストに戻す。

基本的な使い方は「テキストを選択 → ボタンをクリック」見出しやリストはカーソルを置くだけで行全体に適用される。

※メニューバーの‘A’ボタンで表示非表示切替

【文書の切替】

MemoTree

技術・設定

ノード一覧

技術・設定

開発

パーストの動作

AI関係

記事

chtgpt 1強の終了\_260525

Windows

cmdプロンプト

管理者権限

現在の階層でcmdを開く

ターミナル

フォント

諸々ノウハウ

OneDriveの共有設定

図解でわかるOneDriveの共有方法

vbaソースの保護

vbalckunl302 vba保護ツール

vbalckunl\_ホーム

vbalckunl\_保護

vbalckunl\_上級

Excel vbaアプリの配布の仕方

著作権放棄の宣言

クリエイティブ・コモンズ

参照設定と遅延結合

xserver

ssh

common/download/auth.php解説

infoドメインを追加するとしたら

技術・設定へようこそ

作成: 2026/05/20 13:49 更新: 2026/06/06 17:07

編集

パーストの動作はこうなります：

| コピー元                                   | 処理  |
|--|---|
| ブラウザの <code>&lt;pre&gt;</code> コードブロック | パターンA: <code>&lt;pre&gt;</code> をコードブロックに変換 |
| Claudeチャット・Webページ                      | パターンB: Quill標準 (HTMLそのまま)                   |
| メモ帳・NanaTerry (コードらしい)                 | パターンC: コードブロックとして挿入                         |
| 通常テキスト                                 | Quill標準                                     |

【カーソル位置に画像の添付】

画像選択、ドラッグドロップ

表形式の取り込み

MemoTree 常時・汎用

ノード一覧

@temp

- トリコピ用
  - 新しいメモ
  - 表と画像取り込み
  - PDF\_AI導入は進むが...
- トリ保

記事

- TOT
- 用語

ホームページ関係

仕事関係

- 会社googleアカウント

優先順位

- icebreaker
- コーチング\_Todoカード
- ルール守らない
- 会議まとめ
- 生産性を上げる7つの鉄則 イーロン

表と画像取り込み

作成: 2026/05/29 21:05 更新: 2026/06/06 21:04

率直な難易度評価

| 機能         | 難易度        |
|------------|------------|
| ツリー+テキスト編集 | 低~中        |
| 画像の貼り付け    | 中          |
| PDF表示      | 中(ライブラリあり) |
| Webページクリップ | 高          |

Webクリップ(広告除去・整形)が最も難しい部分です。完全自動は難しいので、最初はWebViewで表示してから手動で不要部分を削除するという▼式が現実的です。

各種設定

CSS I/O デバイス レジストリ サウンド ストリーム処理 イベント

一般 FILEモード IFOモード ISO読み込みモード ISO書き込みモード

起動時

- ☒メインムービーPGCを選択
- ☒ストリーム処理を有効
- ☒あいまいなCellを非チェック

オプション

ファイル分割: Chapterごと

☐IFOファイルをコピー

- ☒構造保護を除去
- ☒ROI保護を除去
- ☒RCE保護を除去
- ☒PUOを除去

☒VOB PUOを除去

☒M2Vタイムコード(00:00:00:00)をパッチ

ファイルの追加作成

- ☒Stream Information
- ☐Cell Information - CCE
- ☐Chapter Information - BSPlayer
- ☐Chapter Information - CCE
- ☐Chapter Information - DVDLab
- ☐Chapter Information - DVDMaestr
- ☐Chapter Information - IfoEdit
- ☐Chapter Information - LBA
- ☐Chapter Information - OGG
- ☐Chapter Information - Scenarist
- ☐Vob ID Information - DoltFast4U!

ファイル名

- ☐PGC番号を含める
- ☐アングル番号を含める

デフォルト

OK

キャンセル

[常時・汎用] 保存しました 21:03:19

## 【PDF の取り込み】

PDF の取り込み

The screenshot shows the MemoTree application interface. On the left is a sidebar with a tree view of folders and documents. The main area displays a document titled "PDF\_AI導入は進むが...". The document content includes a disclaimer, a title "AI導入は進むが……IT部門の8割が悲鳴「自社インフラが耐えられない」", a URL, a date "2026年05月01日 13時00分 更新", and several paragraphs of text. A toolbar at the top contains various icons, including a PDF icon. A callout box with the text "PDF の取り込み" points to this PDF icon. The status bar at the bottom indicates "[常時・汎用] 保存しました 21:03:19".

メモリーツリー

常時・汎用

PDF\_AI導入は進むが...

縮小 拡大 等倍 作成: 2026/06/06 21:02 更新: 2026/06/06 21:04

本サービスにおける著作権および一切の権利はアイティメディア株式会社またはその情報提供者に帰属します。また、本サービスの出力結果を無断で複写・複製・転載・転用・頒布等を行うことは、法律で認められた場合を除き禁じます。

シャドーAIは“インフラ整備の遅れ”が原因?:

### AI導入は進むが……IT部門の8割が悲鳴「自社インフラが耐えられない」

<https://atmarkit.itmedia.co.jp/ait/articles/2605/01/news054.html>

Nutanixは企業のITインフラ動向を調査した年次レポート「Enterprise Cloud Index」を公開した。AI活用が広がる中で、多くの企業のインフラがAIワークロードに対応できていない現状が明らかになった。

2026年05月01日 13時00分 更新

[@IT]

Nutanixは2026年3月24日、企業のITインフラ動向を調査した年次レポート「Enterprise Cloud Index」を公開した。同調査は世界14カ国の従業員数500人以上の企業に勤務するITおよびエンジニアリング部門の幹部1600人を対象に、調査会社Wakefield Researchが2025年11月13～23日に実施したものだ。

調査ではITリーダーの85%がAIによってコンテナ化が加速していると回答する一方で、82%は自社のインフラがAIワークロードに十分に対応できていないと回答するなど、AI導入が広がる中での課題感が浮き彫りになった。具体的な内容は以下の通り。

#### AIの普及で変わりつつあるインフラの前提

レポートはAI導入のスピードとインフラの準備状況にギャップが生じているとし、AIは単に新しいワークロードであるだけでなく、インフラの見直しを迫る要因にもなっていると指摘している。

従来のインフラの課題として、レポートはハードウェアやソフトウェアの進化にインフラが追いついていない現状を挙げている。AIの普及を背景に、GPUやAIアクセラレーター、メモリ、サーバ設計などの新技術が短期間のうちに登場する状況となっている。特定のベンダーや構成へのロックインを回避しつつ、新しいハードウェアを柔軟に取り込めるインフラの必要性

[常時・汎用] 保存しました 21:03:19



【メモのロック】

メモのロック

MemoTree \* 常時・汎用

ノード一覧

@temp

- トリコビ用
  - 表と画像取り込み
  - PDF\_AI導入は進むが...
  - purchases テーブル
- とり保
  - SSH
  - square

記事

- TOT
- 用語

ホームページ関係

仕事関係

- 会社googleアカウント
- 優先順位
  - icebreaker
  - コーチング\_Todoカード
  - ルール守らない
  - 会議まとめ
  - 生産性を上げる7つの鉄則 イーロン

新しいメモ

作成: 2026/05/29 21:05 更新: 2026/06/06 21:14 ロック中

Heading 3

B I U

A

検索

ブログ

• こちらも指定なし → RESTRICT ( 同上 )

purchases テーブル

sql

cb:

CONSTRAINT `fk\_purchases\_apps` FOREIGN KEY (`app\_no`) REFERENCES `apps` (`app\_no`) ON DELETE NO ACTION ON UPDATE CASCADE

• purchases.app\_no は必ず apps.app\_no に存在する値でなければならない

• ON DELETE NO ACTION : appsのレコードを削除しようとするとエラーになって阻止される ( RESTRICTと実質同じ )

• ON UPDATE CASCADE : apps.app\_no が変更されたら、 purchases.app\_no も自動的に同じ値に更新される

sql

cb:

CONSTRAINT `fk\_purchases\_user` FOREIGN KEY (`user\_no`) REFERENCES `users` (`user\_no`)

• purchases.user\_no は必ず users.user\_no に存在する値でなければならない

• 存在しないユーザーの購入記録は登録できない

• 指定なし → RESTRICT

まとめ表

| 制約名               | テーブル            | 参照先   | DELETE時          | UPDATE時          |
|-------------------|-----------------|-------|------------------|------------------|
| fk_dl_logs_app    | download_logs   | apps  | RESTRICT ( 阻止 )  | RESTRICT ( 阻止 )  |
| fk_dl_tokens_app  | download_tokens | apps  | RESTRICT ( 阻止 )  | RESTRICT ( 阻止 )  |
| fk_purchases_apps | purchases       | apps  | NO ACTION ( 阻止 ) | CASCADE ( 連動更新 ) |
| fk_purchases_user | purchases       | users | RESTRICT ( 阻止 )  | RESTRICT ( 阻止 )  |

purchases だけ ON UPDATE CASCADE が設定されており、 app\_noが変わっても購入履歴が自動追従するように設計されているのが特徴です。 appsを作り直す際に app\_noの値が変わる

「常時・汎用」を開きました

● 未保存



## 【ノード一覧非表示・書式ツールバー非表示】

The screenshot shows the MemoTree application interface. At the top, there is a toolbar with various icons. Two callout boxes point to specific icons: 'ノード一覧非表示' (Hide Node List) points to the icon representing a list of nodes, and 'ツールバー非表示' (Hide Toolbar) points to the icon representing the toolbar itself. Below the toolbar, the main content area displays a code file named 'SetComment\_Prop.cs 解説'. The code includes a comment about 'InterfaceIsUnknown' and a section titled 'Native 静的クラス (P/Invoke 宣言)'. A third callout box, 'コードブロック内の字下げ' (Indentation in code block), points to the indentation of the code within a code block. The code block contains the following C# code:

```
cb:

internal static class Native
{
    [DllImport("Shell32.dll", CharSet = CharSet.Unicode, PreserveSig = true)]
    public static extern int SHGetPropertyStoreFromParsingName(
        string pszPath,
        IntPtr pbc,
        uint flags,
        ref Guid riid,
        out IPropertyStore ppv);

    [DllImport("Ole32.dll", PreserveSig = true)]
    public static extern int PropVariantClear(ref PROPVARIANT pvar);
}
```

Below the code block, there is a list of bullet points explaining the code:

- SHGetPropertyStoreFromParsingName パス ( pszPath ) から IPropertyStore を取得する Shell API。
- pbc はバインドコンテキスト (ここでは未使用なので IntPtr.Zero )。
- flags は取得方法 (読み書き可能で開く等)。
- riid は「どのインターフェースが欲しいか」を示す IID (ここでは IPropertyStore の IID)。
- 成功すると ppv に IPropertyStore が返る。
- PreserveSig = true により、HRESULT をそのまま int として受け取る (失敗時も例外にしない)。
- PropVariantClearPROPVARIANT の中にあるネイティブ資源 (文字列ポインタ等) を正しく解放する関数。
- 最後に必ず呼んでメモリリークを防ぐ。

Below the list, there is a section titled 'FileCommentWriter クラス' and a comment: 'Explorer の「コメント」プロパティを書き込むユーティリティ。' followed by the start of the FileCommentWriter class definition:

```
cb:

public static class FileCommentWriter
{
```

At the bottom of the window, a status bar shows the message: 「開発」を開きました

【検索:ノード内・文書内・全文書】

検索

The screenshot shows the MemoTree application interface. The main window displays a document titled "SetComment\_Prop.cs 解説". The document content includes a C# code snippet for a static class "Native" and a list of bullet points explaining the code. A search window is open in the foreground, showing the search results for the term "count". The search results list several occurrences of the word "count" in the document, including comments and code. A callout box points to one of the search results, indicating that clicking on the result will display the corresponding text in the main document window.

SetComment\_Prop.cs 解説

作成: 2026/05/31 22:25 更新: 2026/05/31 22:25

- `InterfaceIsIUnknown` は `IUnknown` ベース (一般的な COM インターフェース)。
- 返り値は `HRESULT` 相当 (0 が成功)。 `SetValue` → `Commit` が更新の基本手順。

### Native 静的クラス (P/Invoke 宣言)

cb:

```
internal static class Native
{
    [DllImport("Shell32.dll", CharSet = CharSet.Unicode)]
    public static extern int SHGetPropertyStoreFromParsingName(
        string pszPath,
        IntPtr pbc,
        uint flags,
        ref Guid riid,
        out IPropertyStore ppv);

    [DllImport("Ole32.dll", PreserveSig = true)]
    public static extern int PropVariantClear(ref PROPVARIANT pv);
}
```

- `SHGetPropertyStoreFromParsingName` パス ( `pszPath` ) から
- `pbc` は バインド コンテキスト (ここでは未使用なので)
- `flags` は 取得方法 (読み書き可能で開く等)。
- `riid` は 「どのインターフェースが欲しいか」を示す GUID
- 成功すると `ppv` に `IPropertyStore` が返る。
- `PreserveSig = true` により、`HRESULT` をそのまま返す。
- `PropVariantClear` の中にあるネイティブ関数を呼び出す。
- 最後に必ず呼んでメモリリークを防ぐ。

### FileCommentWriter クラス

Explorer の「コメント」プロパティを書き込むユーティリティ。

cb:

```
public static class FileCommentWriter
{
    // ...
}
```

「開発」を開きました

検索

検索文字列: count

実行

範囲: ☐ 指定ノード以下 ☐ 現在の文書 ☒ すべての文書

[常時・汎用] 無料版・有料版ダウンロード : 現状 /home/xs441478/e-joshis.com/private\_downlo  
[常時・汎用] 無料版・有料版ダウンロード : head項目 'account' → 無料版zipファイル名 'account.z  
[常時・汎用] 無料版・有料版ダウンロード : head項目 'account' → 有料版zipファイル名 'account\_1  
[開発] SetComment\_Prop.cs 解説 : uint GetCount(out uint cProps);  
[開発] xamppからxserverヘデータを移行 : SELECT COUNT(\*) FROM shohin\_mst; -- 14件になる;  
[開発] コードブロックがテーブルになる : Sheets.Add After:=Sheets(Sheets.Count)  
[開発] 今日のメモ : FileInt = FSO.GetFolder(FolderPath).Files.Count '指定したフォルダ内のファイル  
[開発] v20\_ブログコピー&セクション対応 : int startHtml = utf8.GetByteCount(string.Format(h  
[開発] v20\_ブログコピー&セクション対応 : int startFragment = startHtml + utf8.GetByteCount(p  
[開発] v20\_ブログコピー&セクション対応 : int endFragment = startFragment + utf8.GetByteCo  
[開発] v20\_ブログコピー&セクション対応 : int endHtml = endFragment + utf8.GetByteCoun  
[開発] v34\_画像、RTF対応 : if (RootNodes.Count > 0)  
[開発] v34\_画像、RTF対応 : SelectedNode = RootNodes[0].Children.Count > 0  
[開発] 右クリックメニュー設定 ori : if (\$pos -ge 0 -and \$pos -le \$chars.Count) {  
[開発] 右クリックメニュー設定 ori : Write-Host ("Converted: {0} -> {1} (boundaries: {2})" -f \$inPa

65 件ヒットしました。行を選ぶと該当箇所を表示します。

検索結果欄のメモをクリックすると該当箇所がメモ欄に表示される

## 【設定】

The screenshot shows the MemoTree application interface. The main window displays a code editor with C# code for `SetComment_Prop.cs`. The code defines a static class `Native` with two methods: `SHGetPropertyStoreFromParsingName` and `PropVariantClear`. Below the code, there are several bullet points explaining the code's functionality. A settings dialog box is open in the foreground, titled "設定" (Settings). The dialog has a close button (X) in the top right corner. It contains the following sections:

- 表示 (Display):**
  - フォント (Font): `Plemon Console NF`
  - 文字サイズ (Text Size): `標準 (14px)`
  - テーマ (Theme): `ダークモードに切替` (Switch to Dark Mode)
- 動作 (Action):**
  - 自動保存 (Auto Save): ☐ 有効 (Enabled)
- 文書名の変更 (Change Document Name):**
  - 文書1 (常時・汎用) (Document 1 (Always/General)): `常時・汎用`
  - 文書2 (技術・設定) (Document 2 (Technical/Settings)): `技術・設定`
  - 文書3 (開発) (Document 3 (Development)): `開発`
- データ管理 (Data Management):**
  - 印刷プレビュー (Print Preview): `印刷プレビュー`

At the bottom of the dialog, there are two buttons: `適用して閉じる` (Apply and Close) and `キャンセル` (Cancel).

The background code editor shows the following C# code:

```
cb:

internal static class Native
{
    [DllImport("Shell32.dll", CharSet = CharSet.Unicode, PreserveSig = false)]
    public static extern int SHGetPropertyStoreFromParsingName(
        string pszPath,
        IntPtr pbc,
        uint flags,
        ref Guid riid,
        out IPropertyStore ppv);

    [DllImport("Ole32.dll", PreserveSig = true)]
    public static extern int PropVariantClear(ref PROPVARIANT pvar);
}
```

Below the code, there are several bullet points explaining the code's functionality:

- `SHGetPropertyStoreFromParsingName` パス ( `pszPath` ) から `IPropertyStore` を取得する。
- `pbc` はバインドコンテキスト (ここでは未使用なので `IntPtr.Zero` )。
- `flags` は取得方法 (読み書き可能で開く等)。
- `riid` は「どのインターフェースが欲しいか」を示す IID (ここでは `IID_IPropertyStore` )。
- 成功すると `ppv` に `IPropertyStore` が返る。
- `PreserveSig = true` により、`HRESULT` をそのまま `int` として受け取る (失敗時も例外にしない)。
- `PropVariantClear` の中にあるネイティブ資源 (文字列ポインタ等) を正しく解放する関数。
- 最後に必ず呼んでメモリリークを防ぐ。

Below the code, there is a section titled **FileCommentWriter クラス** (FileCommentWriter Class). It describes the class as a utility for writing properties to Explorer's "Comments" property. The code for the class is as follows:

```
cb:

public static class FileCommentWriter
{
    // ...
}
```

以上