Windows XP の視覚スタイルおよび PrintWindow を Windows アプリケーションで使用する

Paul Hellyar
Microsoft Corporation

October 25, 2001
日本語版最終更新日 2002 年 11 月 12 日

この記事で使用されているサンプル アプリケーションは、MSDN Downloads [http://msdn.microsoft.com/ja-jp/downloads/default.aspx] からダウンロードできます。

> **注：**このサンプルは、Windows XP を実行するマシン上でのみ機能します。サンプルを開くと、複数のアプリケーション ウィンドウが開きます。AltTab を使用して TaskSwitcher アプリケーションを実行してください。

要約：この記事では、Windows XP の新しい視覚スタイルおよび PrintWindow がどのように Windows アプリケーションで使用できるかを示すデモのフレームワークとして、より重要性を増した AltTab アプリケーション、TaskSwitcher を皆さんと共に開発していきます。

目次

はじめに

MicrosoftR WindowsR XP では、使いやすい優れたユーザー インターフェイスを提供する、新しい視覚スタイルが採用されています。この新しいユーザー インターフェイスには、丸みをおびたウィンドウ、より使いやすくなったタスク バー、マウスでポイントされたときにフォーカスされる UI 要素などが含まれます。

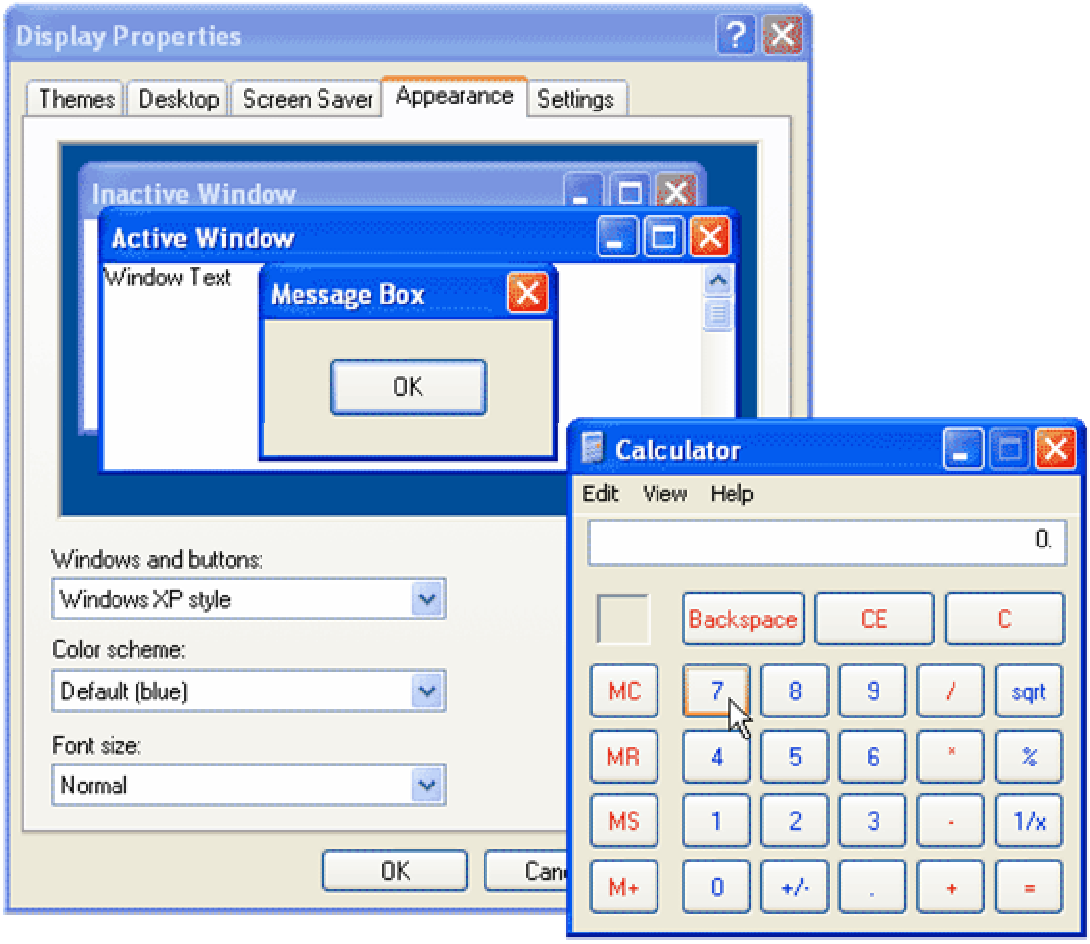

図 1. 新しい視覚スタイルでの電卓と [画面のプロパティ] ダイアログ ボックス

また、Windows XP では新しい印刷 API である PrintWindow [http://msdn2.microsoft.com/en-us/library/ms535695.aspx] も採用されました。この API により、印刷元は、ウィンドウの視覚コピーのスナップショットをデバイス コンテキストに作成することができます。

視覚スタイル、およびアプリケーションに視覚スタイルを適用するための詳細については、MSDN ライブラリの記事「Windows XP ビジュアル スタイルの使用」を参照してください。「Windows XP ビジュアル スタイルの使用」では一般的な概要や情報を提供していますが、この記事では、新しい視覚スタイル API および PrintWindow API の実用例を紹介します。また、既存の Win32R API の使用についても多少触れています。

この記事では、以前のバージョンの Windows オペレーティング システムで提供されていた AltTab メカニズムと同様の機能を提供する TaskSwitcher アプリケーションを開発していきます。このアプリケーションでは、アイコンの一覧を表示するほかに、切り替えの対象となるアプリケーションの実寸表示プレビューも表示されます。アプリケーション アイコンおよびプレビューが表示されるコンテナ ウィンドウにも新しい視覚スタイルを採用し、エンド ユーザーが選択した視覚スタイルの外観とアプリケーションを合わせることができるようにします。

ページのトップへ

TaskSwitcher アプリケーション

TaskSwitcher は、Windows XP に既存の AltTab アプリケーション切り替えメカニズムを置き換えることができるように設計されています。AltTab は、Windows に組み込まれているパワー ユーザー向けの機能です。この機能を使用することにより、エンド ユーザーは上位レベルのアプリケーション ウィンドウをすばやく切り替えることができます。ユーザーが **Alt** + **Tab** キーを押すと、現在開かれているウィンドウの一覧が Windows によって作成されます。この一覧では、開かれているウィンドウがアイコンとして表示されます。現在選択しているウィンドウを示すアイコンには、アイコンの周りに四角形の枠が表示されます。**Alt** キーを押したまま **Tab** キーを押すと、四角形の枠が次のアイコンに移動します。四角形の枠で囲まれているアイコンは、エンド ユーザーが **Alt** キーを離したときに Windows よって画面の前面に表示されるアプリケーションを表しています。

図 2. Windows XP の AltTab コンテナ ウィンドウ

この機能は、論理的に 3 つのコンポーネントに分類することができます。まず、アプリケーションは **Alt** + **Tab** というキーの組み合わせをリッスンする必要があります。アプリケーションがこのキーの組み合わせを受け取ると、デスクトップ上にある上位レベルのアプリケーション ウィンドウを列挙します。最後に、これらのウィンドウを一種の UI コンテナ内に表示し、ユーザーが目的のアプリケーションを示すアイコンを選択できるようにします。

ページのトップへ

キーボード入力を中断する

Win32 API を使用することにより、特定のキーボード操作 (キーストローク) をリッスンするアプリケーションを作成できます。キーボード リスナの実装方法はいくつかありますが、最も簡単なのは API RegisterHotKey [http://msdn2.microsoft.com/en-us/library/ms646309.aspx] を使用する方法です。この API は "hwnd"、"ID"、"Virtual Key"、および "Key Modifier" (補助キー) を受け取ります。呼び出しが成功した場合は、Virtual Key と Key Modifier が押されたときに、hwnd の WndProc が ID と同等のメッセージの wParam を含む WM_HOTKEY メッセージを受け取ります。この動作は、リスナ アプリケーション ウィンドウがアクティブであるかどうかにかかわらず起こります。次の呼び出しを行うと、**Alt** + **Tab** が押されたときに、hwndApp が WM_HOTKEY メッセージで通知されるようになります。

コードのコピー

```
RegisterHotKey(hwndApp, IDH_ALTTAB, MOD_ALT, VK_TAB)
```

Windows XP より前の Windows オペレーティング システムでは、AltTab をホット キーとして登録しようとするとエラーが発生しました。Windows XP ではこれが可能になり、さらに、ユーザーはこのイベントに対して Windows XP の組み込み AltTab ホット キー ハンドラを使用するのではなく、独自の処理を施すことができるようになりました。

コードのコピー

```
// ホット キーをリッスンするダミー ウィンドウを作成します。
HWND hwndApp = CreateWindow(WC_APP, NULL, 0, 0, 0, 0, 0, HWND_MESSAGE,
               NULL, THIS_EXE, NULL);
if (hwnd)
{
    // Alt + Tab キーを登録します。
    RegisterHotKey(hwndApp, IDH_NEXT, MOD_ALT, VK_TAB);
    RegisterHotKey(hwndApp, IDH_PREV, MOD_ALT|MOD_SHIFT, VK_TAB);

    MSG msg;
    while (GetMessage(&msg, NULL, 0, 0))
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
}

LRESULT WndProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam)
{
    switch (uMsg)
    {
        case WM_HOTKEY:
        {
            switch (wParam)
            {
                // コンテナ ウィンドウに表示されない場合は、上位レベルの
                // ウィンドウを列挙してそれらのアイコンとテキストを抽出し、
                // コンテナ ウィンドウに表示します。
                case IDH_NEXT:
```

```
                {
                    // ウィンドウ階層内の次の上位レベル ウィンドウ
                    // を示すアイコンを選択します。
                    break;
                }
                case IDH_PREV:
                {
                    // ウィンドウ階層内の前の上位レベル ウィンドウ
                    // を示すアイコンを選択します。
                }
            }
        }
    }
}
```

キーボード リスナを実装するためのより高度な方法としては、API SetWindowsHookEx [http://msdn2.microsoft.com/en-us/library/ms644990.aspx] と WH_KEYBOARD_LL を使用する方法があります。この方法では、現在のデスクトップに下位レベルのキーボード フック レイヤをグローバルに作成します。SetWindowsHookEx 関数で指定されている LowLevelKeyboardProc コールバック関数はすべてのキーボード入力を受け取ります。キーボード入力の処理後、LowLevelKeyboardProc は次のフック チェイン (ほとんどの場合は対象アプリケーション) でこのキーボード入力を受け取るようにするために、CallNextHookEx を呼び出します。LowLevelKeyboardProc はすべてのキーボード入力を受け取るため、**Alt** + **Tab** キーの組み合わせをリッスンする安定したメカニズムとして簡単に実装することができます。アプリケーションが独自の AltTab メカニズムを実装する場合は、この時点でウィンドウ列挙アルゴリズムが実行され、AltTab キー イベントを送らずに LowLevelKeyboardHook から戻ります。

コードのコピー

```
hhook = SetWindowsHookEx(WH_KEYBOARD_LL, LowLevelKeyboardProc, hinst, 0);

LRESULT LowLevelKeyboardProc(INT nCode, WPARAM wParam, LPARAM lParam)
{
    static BOOL fShiftPressed = FALSE;

    BOOL fHandled = FALSE;

    if (nCode == HC_ACTION)
    {
        KBDLLHOOKSTRUCT *pkbdllhook = (KBDLLHOOKSTRUCT *)lParam;

        switch (wParam)
        {
            case WM_SYSKEYDOWN:
                switch (pkbdllhook->vkCode)
                {
                    case VK_LSHIFT:
                    case VK_RSHIFT:
                    {
                        // ユーザーが Shift キーを押しました。
                        fShiftPressed = TRUE;
                        break;
                    }
                    case VK_TAB:
                    {
                        if (pkbdllhook->flags & LLKHF_ALTDOWN)
                        {
                            // ユーザーが Alt + Tab キーを押したので、
                            // AltTab ホット キー ハンドラを実行します。
                            fHandled = TRUE;
                        }
                        break;
                    }
                    case VK_ESCAPE:
                    {
                        if (pkbdllhook->flags & LLKHF_ALTDOWN)
                        {
                            // ユーザーが Esc キーを押したので、選択されたウィンドウに
                            // 切り替えずに、AltTab コンテナ ウィンドウを終了します。
                            fHandled = TRUE;
                        }
                        break;
                    }
                }
```

```
                break;

            case WM_KEYUP:
            case WM_SYSKEYUP:
                switch (pkbdllhook->vkCode)
                {
                    case VK_LMENU:
                    case VK_RMENU:
                    {
                        // ユーザーが Alt キーを離したので、選択されたウィンドウに
                        // 切り替えて、AltTab コンテナ ウィンドウを終了します。
                        break;
                    }
                    case VK_LSHIFT:
                    case VK_RSHIFT:
                    {
                        // ユーザーが Shift キーを離しました。
                        fShiftPressed = FALSE;
                        break;
                    }
                }

                break;
        }
    }

    return (fHandled ? TRUE : CallNextHookEx(hhook, nCode, wParam, lParam));
}
```

ページのトップへ

上位レベルのアプリケーション ウィンドウを列挙する

上位レベルのアプリケーション ウィンドウの列挙は、Win32 API EnumWindows [http://msdn2.microsoft.com/en-us/library/ms633497.aspx] を使用することによって簡単に行うことができます。これは優れた API であり、パラメータとして EnumFunc コールバック関数を受け取ります。デスクトップ上の上位レベルの各ウィンドウについて、システムにより、上位レベル ウィンドウのウィンドウ ハンドルをパラメータとして含む EnumFunc がコールバックされます。すべての上位レベル ウィンドウが AltTab の一覧に表示されるわけではありません。ウィンドウの多くのプロパティに対してクエリが実行され、さまざまな条件に一致するもののみが選ばれます。条件には、ウィンドウがアプリケーション ウィンドウであるか、ウィンドウをアクティブにすることは可能か、表示可能か、ToolWindow かどうか、などがあります。

AltTab イベントを受け取ると、TaskSwitcher は EnumWindows を使用してデスクトップ上の上位レベル ウィンドウを列挙し始めます。システムは、各上位レベル ウィンドウについてこの関数をコールバックします。すべての条件を満たすウィンドウは、AltTab の一覧に含まれるウィンドウとして加えられます。

ページのトップへ

上位レベルのアプリケーション ウィンドウを表示する

AltTab の一覧に表示されている UI では、TaskSwitcher は Windows XP の新しいプログラミング機能を多数使用します。選択したアプリケーションのテキストには新しい API DrawShadowText が適用されます。ウィンドウによるプレビューは、新しい API PrintWindow を使用して生成されます。また、おそらくアプリケーション開発者にとって最も関心のある事実は、TaskSwitcher は Windows XP の新しい視覚スタイルを使用するということでしょう。

ウィンドウの情報を収集する

AltTab の一覧に表示するウィンドウの一覧が生成されると、各ウィンドウのさまざまな属性が取得され、プレビューコンテナで適用されます。各ウィンドウのアイコンは、各ウィンドウに WM_GETICON [http://msdn2.microsoft.com/en-us/library/ms632625.aspx] ウィンドウ メッセージが送られることによって認識され、一覧表示されます。ユーザーが **Tab** キーを押して一覧内の項目を切り替えると、選択されたアプリケーションのアイコンとテキストがプレビュー コンテナ上部に表示されます。私は、API GetWindowText [http://msdn2.microsoft.com/en-us/library/ms633520.aspx] を使用して、各ウィンドウのキャプション テキストを取得しました。アプリケーション テキストの使用について興味深い点として、アプリケーションでは新しい API comctl32 v6 API DrawShadowText [http://msdn2.microsoft.com/en-us/library/ms672637.aspx] を使用するということが挙げられます。 Windows XP におけるこの新規 API は、API DrawText が受け取るすべてのパラメータと、テキストと影の色、および影の x/y オフセットを示す 2 つの COLORREF を受け取ります。

ウィンドウ プレビューに色を付ける

TaskSwitcher は、選択したウィンドウの実寸表示プレビューも提供します (選択したウィンドウが最小化されている場合を除きます - その場合はウィンドウのタイトル バーのみが表示されます)。実寸表示プレビューに色を付ける際、TaskSwitcher は、ダブル バッファリングやハーフトーン スケーリングなど、Win32 の高レベルのペインティング技術を使用します。ただし、ウィンドウ プレビューの取得には新しい Windows XP user32 API PrintWindow が使用されます。PrintWindow は "ウィンドウ ハンドル"、"hdc"、および "予約済みフラグ" を受け取ります。この API は、ウィンドウのスナップショットの色付けに、hdc へのウィンドウ リダイレクションを使用します。

コードのコピー

```
// メモリ デバイス コンテキスト hdcMem に格納されている、
// ウィンドウ hwnd のスナップショットを受け取ります。
HDC hdc = GetWindowDC(hwnd);
if (hdc)
{
    HDC hdcMem = CreateCompatibleDC(hdc);
    if (hdcMem)
    {
        RECT rc;
        GetWindowRect(hwnd, &rc);

        HBITMAP hbitmap = CreateCompatibleBitmap(hdc, RECTWIDTH(rc), RECTHEIGHT(rc));
        if (hbitmap)
        {
            SelectObject(hdcMem, hbitmap);

            PrintWindow(hwnd, hdcMem, 0);

            DeleteObject(hbitmap);
        }
        DeleteObject(hdcMem);
    }
    ReleaseDC(hwnd, hdc);
}
```

視覚スタイル API をコンテナに適用する

コンテナ ウィンドウに含まれるアイテムにはすべて色が付きます。コンテナ ウィンドウのバックグラウンドは Windows XP の新しい視覚スタイルに対応しています。これは、丸みをおびたウィンドウや、タイトル バーのバックグラウンドのような色の変化具合を持つバックグラウンドなど、Windows XP の項目と同じ外観を持つことを意味します。コンテナのバックグラウンドに視覚スタイルを提供する際、TaskSwitcher は uxtheme.h に含まれている多くのテーマ API を使用します。これには、OpenThemeData [http://msdn2.microsoft.com/en-us/library/ms649963.aspx]、CloseThemeData [http://msdn2.microsoft.com/en-us/library/ms649807.aspx]、GetThemeBackgroundRegion [http://msdn2.microsoft.com/en-us/library/ms649927.aspx]、DrawThemeBackground [http://msdn2.microsoft.com/en-us/library/ms649808.aspx] が含まれます。この例では、コンテナ ウィンドウのバックグラウンドとして、スタート パネル上部に使用されている視覚スタイルを適用します。

コードのコピー

```
#include <uxtheme.h>
#include <tmschema.h>

// AltTab の一覧コンテナ ウィンドウのためのダイアログ プロシージャです。
INT_PTR CALLBACK DlgProc(HWND hwnd, UINT uMsg, WPARAM, LPARAM lParam)
{
    static HTHEME htheme = NULL;

    switch (uMsg)
    {
        case WM_INITDIALOG:
        {
            htheme = OpenThemeData(hwnd, L"StartPanel");

            if (htheme)
            {
                // コンテナ ウィンドウに適用する部分のバックグラウンド領域を取得し、
                // ダイアログに適用します。
                HRGN hrgn = NULL;
                GetWindowRect(hwnd, &rc);
```

```
                OffsetRect(&rc, -rc.left, -rc.top);

                if (SUCCEEDED(GetThemeBackgroundRegion(htheme, NULL,
                            SPP_USERPANE, 0, &rc, &hrgn)))
                {
                    SetWindowRgn(hwnd, hrgn, FALSE);
                }
            }

            break;
        }
        case WM_PAINT:
        {
            PAINTSTRUCT ps;
            HDC hdc = BeginPaint(hwnd, &ps);
            if (hdc)
            {
                if (htheme)
                {
                    // 視覚スタイルはアクティブな状態です。API を使用して色付けを
                    // 行います。

                    RECT rc;
                    GetWindowRect(hwnd, &rc);
                    OffsetRect(&rc, -rc.left, -rc.top);

                    DrawThemeBackground(htheme, hdc, SPP_USERPANE, 0, &rc, NULL);
                }
                else
                {
                    // 視覚スタイルは非アクティブな状態です。従来のウィンドウ スタイルで
                    // 色付けを行います。
                }
            }
            EndPaint(hwnd, &ps);

            break;
        }
        case WM_THEMECHANGED:
        {
            // 視覚スタイルが変更されました。既存の htheme を閉じ、
            // 新しい htheme を開いてみます。
            if (htheme)
            {
                CloseThemeData(htheme);
            }
            htheme = OpenThemeData(hwnd, L"StartPanel");

            break;
        }
    }
}
```

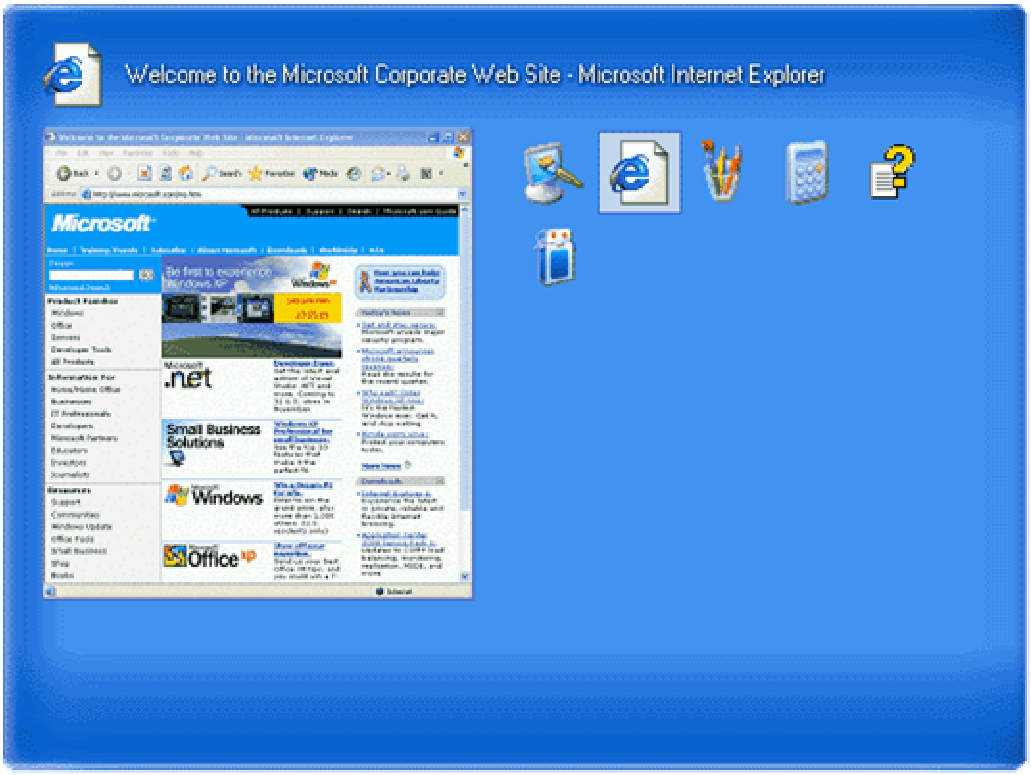

図 3. TaskSwitcher AltTab コンテナ ウィンドウ

ページのトップへ

Comctl32.dll Version 6 を使用する

Taskswitcher は、Comctl32.dll Version 6 の新機能のいくつかを利用します。たとえば、アイコンの一覧は、ListView コントロールと、コンテナのバックグラウンドと一致するバックグラウンド ウォーターマークを使用して実装されます。これにより、アイコンの一覧はウィンドウ内のその他の領域と一体化して表示されます。さらに、Comctl32.dll Version 6 では API DrawShadowText も提供されています。

Comctl32 Version 6 はサイド バイ サイド DLL です。これは、Comctl32.dll Version 6 と Version 5 の両方を 1 つのシステム上に共存させることができることを意味します。既定では、アプリケーションが静的に Comctl32.lib にリンクされる場合、アプリケーションは Version 5 を使用します。アプリケーションで Version 6 を利用するには、次のようなマニフェスト ファイルをアプリケーションに提供する必要があります。

コードのコピー

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion="1.0">
<assemblyIdentity
version="1.0.0.0"
processorArchitecture="X86"
name="Microsoft.Shell.TaskSwitch "
type="win32"
/>
<description>AltTab の代替項目を切り替えます。</description>
<dependency>
<dependentAssembly>
<assemblyIdentity
type="win32"
name="Microsoft.Windows.Common-Controls"
version="6.0.0.0"
processorArchitecture="X86"
publicKeyToken="6595b64144ccf1df"
language="*"
/>
</dependentAssembly>
</dependency>
</assembly>
```

.rc ファイル内の次の行を指定することで、マニフェスト ファイルは、アプリケーションのリソース セクションにコンパイルされます。

コードのコピー

```
CREATEPROCESS_MANIFEST_RESOURCE_ID RT_MANIFEST "TaskSwitch.exe.manifest"
```

ページのトップへ

結論

Windows XP は、新しい視覚スタイル、ウィンドウのコンテンツを視覚的にキャプチャする機能など、根本的に設計し直されたユーザー インターフェイスを提供します。この記事で紹介したテクニックを使用することによって、開発者は視覚スタイル API を活用して、ユニークでありながらも Windows XP と同様の外観を持つ独自のアプリケーションを作成することができます。PrintWindow を使用すると、開発者は任意のウィンドウの視覚コピーをデバイス コンテキストに作成することができます。

ページのトップへ