

한글 Addon Action

Event Handler 추가하기

소 속	한글과컴퓨터
문서번호	
버 전	1.0 (외부 배포용)
일 자	2008. 08. 04

Document Info

문서 목표	한글의 Addon Action을 이해하고 실행하는 법을 익힌다.
문서 요약	<ol style="list-style-type: none"> 1. 개요 2. Event Handler 추가방법 설명 3.

Revision History

Version	Date	Author	Reason For Changes	Update Log
1.0	2008-08-04	오흥석	최초 작성	

User Action Module

1. User Action Module 추가 방법

가. 모듈 등록

User Action Module을 사용하기 위해서는 해당 모듈이 먼저 레지스트리에 등록되어 있어야 합니다.

레지스트리의 HKEY_CURRENT_USER\Software\Hnc\HwpUserAction\Modules에 등록합니다.

ex) TestUserActionModule.dll 파일을 "TestUserAction" 이라는 모듈명으로 등록



나. DLL 구현

[UserActionDLL]로 등록되는 DLL은 아래 함수를 Export 함수로 가져야 합니다.

```
interface IHncUserActionModule
{
    virtual LPCTSTR EnumAction(int nIterator) = 0;
    virtual BOOL GetActionImage(IN LPCTSTR szAction, IN UINT uState, OUT HBITMAP* lphBitmap, OUT int* lpnImageIndex) = 0;
    virtual BOOL UpdateUI(IN LPCTSTR szAction, LPDISPATCH pObject /*IHwpObject*/, OUT UINT* lpuState) = 0;
    virtual int DoAction(LPCTSTR szAction, LPDISPATCH pObject /*IHwpObject*/) = 0;
};
__declspec(dllexport) IHncUserActionModule* __stdcall QueryUserActionInterface();
```

IHncUserActionModule 는 사용자 액션을 구현한 인터페이스로 아래 함수를 가집니다.

LPCTSTR EnumAction(int nIterator) :

이 함수는 IHwpObject.RegisterModule("UserActionDLL", ...) 으로 등록이 될 때 호출이 되며, 사용자 Action의 이름과 개수를 판별하기 위해 사용됩니다.

nIterator에는 0부터 1씩 증가하면서 호출 됩니다. 구현한 Action의 이름들을 차례로 리턴해야 합니다. 구현한 Action이 더 이상 없을 때는 NULL을 리턴하십시오.

BOOL GetActionImage(IN LPCTSTR szAction, IN UINT uState, OUT HBITMAP* lphBitmap, OUT int* lpnImageIndex):

이 함수는 툴바 버튼을 그릴 때 사용하는 이미지에 대한 정보를 얻을 때 호출 됩니다.

szAction : 사용자 Action 이름

uState : 현재 툴바의 상태 (DSCS_CHECKED, DSCS_HOT, DFCS_INACTIVE, DFCS_PUSHED : 자세한 사항은 Win32 API의 DrawFrameControl 참조)

lphBitmap : HBITMAP형식의 비트맵의 핸들 (비트맵의 형식에 대해서는 Toolbar Button 이미지 형식을 참조)을 저장해야 합니다.

lpnImageIndex : bitmap에서 몇 번째에 위치한 그림인지를 저장합니다.

!주의! : 이 함수는 매우 자주 호출 됩니다. 사용할 HBITMAP 등은 생성자에서 미리 로드해놓기를 권장합니다.

BOOL UpdateUI(IN LPCTSTR szAction, LPDISPATCH pObject /*IHwpObject*/, OUT UINT* lpuState)

이 함수는 사용자 Action의 UI 상태에 대한 정보를 얻을 때 호출됩니다.

szAction : 사용자 Action 이름

pObject : IHwpObject의 인터페이스

lpuState : UI 상태

0 : 보통상태

DFCS_INACTIVE : 비활성 상태

DFCS_CHECKED : 체크된 상태

DFCS_BUTTON3STATE : 중간 상태 (INDETERMINATE)

(DFCS_CHECKED와 DFCS_BUTTON3STATE는 함께 사용할 수 없습니다.)

int DoAction(LPCTSTR szAction, LPDISPATCH pObject /*IHwpObject*/)

이 함수는 사용자 Action이 실행되었을 때 호출됩니다.

szAction : 사용자 Action 이름

pObject : IHwpObject의 인터페이스

2. Toolbar Button 추가 방법

가. Toolbar Button 추가

한글2007에 툴바를 추가하기 위해서는 OnInitialLoad()와 OnLoad()를 정의해 주어야 합니다.

OnInitialLoad()는 User Action Module이 처음 레지스트리에 등록될 때 한번만 실행되는 함수입니다.

UserAction이 등록될 때 한번만 실행되는 루틴을 이곳에 추가합니다.

툴바의 등록작업 역시 한번만 수행하면 되므로 대부분 OnInitialLoad()에서 툴바를 등록하게 됩니다.

OnLoad()는 한글창이 새로 로드될 때마다 실행되는 함수입니다.

한글창이 로드될 때마다 실행되어야 할 루틴을 이곳에 추가합니다.

한글은 새 창을 로드할 때 기본 툴바의 레이아웃을 불러옵니다.

사용자가 등록한 툴바의 레이아웃을 사용하기 위해서 OnLoad()에서 등록된 툴바 레이아웃을 불러오는 작업을 수행해야 합니다.

또한 레이아웃의 변경사항을 체크할 수 있는 IsNewSerializePath() 함수를 사용해서 툴바의 등록작업을 수행할 수도 있습니다.

ToobBar 관련 작업을 하기 위해서는 먼저 Toolbar layout 을 얻어 와야 합니다.

IXHwpToolbarLayout은 IXHwpWindow 개체로부터 얻어올 수 있습니다.

한글의 원래 Layout은 그대로 놔두고 따로 Layout 상태를 보존하기 위해서 ChangeSerializePath()를 사용하여 상태를 저장할 곳을 새로 지정합니다. 만약 이전에 지정된 적이 있다면, 마지막으로 저장된 Layout 상태를 불러옵니다

IsNewSerializePath()를 사용하여 이전에 Layout 상태가 저장된 적이 있는지 판별할 수 있습니다. 처음 지정되어 저장된 정보가 없는 경우에는 TRUE를 리턴합니다.

기존에 추가된 버튼을 또 추가하는 일이 없도록, IsNewSerializePath를 사용하여 이전에 저장된 적이 없을 때만 Layout을 변경하도록 코드를 작성하십시오. Layout을 변경하는 코드가 변경되었을 때는 사용자에게 새로 변경된 Layout이 적용될 수 있도록 ChangeSerializePath에 사용한 이름을 새로운 이름으로 변경하십시오.

ex)

```

BOOL OnInitialLoad(CHwpObject& rHwpObject)
{
    // Toolbar Button을 등록하기 위해서 CXHwpToolBarLayout을 얻어온다.
    CXHwpWindows windows = m_xhwpobject.get_XHwpWindows();
    CXHwpWindow activewindow = windows.get_Active_XHwpWindow();
    CXHwpToolBarLayout tlayout = activewindow.get_XHwpToolBarLayout();

    // 한글의 프레임 상태가 변하지 않도록 SerializePath를 변경한다.
    tlayout.ChangeSerializePath(_T("TestGroupWWProuct1_v1"));

    // 처음으로 SerializePath가 생성되었을 때만 버튼을 추가한다.
    if (tlayout.IsNewSerializePath()) {
        CXHwpToolBar toolbar = tlayout.CreateToolBar(_T("TestToolBar"), 0);

        CXHwpToolBarButton toolbarbutton1 = tlayout.CreateToolBarButton(_T("UserAction1"), UUIDSTR_USERACTION1, 0);
        CXHwpToolBarButton toolbarbutton2 = tlayout.CreateToolBarButton(_T("UserAction2"), UUIDSTR_USERACTION2, 0);
        CXHwpToolBarButton toolbarbutton3 = tlayout.CreateToolBarButton(_T("UserAction3"), UUIDSTR_USERACTION3, 0);

        toolbar.InsertToolBarButton(toolbarbutton1, -1);
        toolbar.InsertToolBarButton(toolbarbutton2, -1);
        toolbar.InsertToolBarButton(toolbarbutton3, -1);
    }

    return TRUE;
}

BOOL OnLoad(CHwpObject& rHwpObject)
{

```

```
// Toolbar Button을 등록하기 위해서 CXHwpToolBarLayout을 얻어온다.  
CXHwpWindows windows = m_xhwpobject.get_XHwpWindows();  
CXHwpWindow activewindow = windows.get_Active_XHwpWindow();  
CXHwpToolBarLayout tlayout = activewindow.get_XHwpToolBarLayout();  
  
// 한/글의 프레임상태를 Path에 정의된 프레임으로 변경한다  
tlayout.ChangeSerializePath(_T("TestGroupWWProuct1_v1"));  
}
```

참고:

- Layout이 저장된 곳은 한글2007의 경우 HKEY_CURRENT_USER\Software\Hnc\Hwp\7.0\HwpFrame\AutomationClient 의 하위키입니다.

나. Toolbar Button 이미지 형식

Toolbar Button에 사용하는 이미지는 16x16 크기의 각각의 버튼이 가로로 60개가 나열된 32bit .BMP 파일입니다.

한글2007이 설치된 HncWHwp70WButtonsWDefault 폴더에서 동일한 이미지 형식들을 확인해 볼 수 있습니다.

Image Format: BMP

Color Depth : 32bit (Red, Green, Blue, Alpha) : 투명색을 포함

Size : Width : 16 x 60 = 960 pixel

Height : 16 x n : 16의 배수

Pixel Format : Source Alpha Blending이 가능하도록 값이 조정되어 있어야 합니다.

참고 : Source Alpha Blending

투명도가 곱해진 값을 BMP에 저장해서 그리는 속도를 조금이라도 빠르게 만든 것이 Source Alpha Blending 기법입니다.

일반적인 Alpha Blending 공식:


$$\text{Pixel_Color} = \text{이미지색} * \text{Alpha} / 255 + \text{배경색} * (255 - \text{Alpha}) / 255$$

프로그램이 실행하는 동안 이미지와 Alpha값이 변경되는 경우는 없다고 가정하고, 이미지색*Alpha/255를 Source-Alpha pixel(SAP)라고 하면, SAP를 상수로 볼 수 있습니다.

Source Alpha Blending 공식:

$$\text{Pixel_Color} = \text{SAP} + \text{배경색} * (255 - \text{Alpha}) / 255$$

ex)

파란색 배경에 노란색 BMP를 Alpha를 오른쪽으로 감소시키며 그렸을 때			
일반 Alpha Blending (R, G, B, A)	(255, 255, 0, 255)	(255, 255, 0, 128)	(255, 255, 0, 0)
Source Alpha Blending (R, G, B, A)	(255, 255, 0, 255)	(128, 128, 0, 128)	(0, 0, 0, 0)

Source Alpha Blending에 사용하는 픽셀의 R, G, B값들은 절대 A 값보다 크면 안됩니다.

만약 A값보다 큰 값이 들어온 경우에는 배경과 혼합되었을 때, 색상 손실이 발생합니다. (일반적으로 255보다 큰 값은 255로 간주하기 때문에 혼합된 부분이 밝게 보입니다.)

3. Toolbar 관련 개체

가. IXHwpToolbarLayout

툴바나 툴바 버튼의 생성/삭제와 관련된 작업을 한다.

1) BOOL DeleteToolbar(BSTR name)

툴바를 삭제한다. 툴바를 잠시 감출 경우는 ShowToolbar를 사용한다.

name : 툴바 이름 (예:그리기, 표, 메모)

2) BOOL IsNewSerializePath()

ChangeSerializePath 이후 새로운 Serialize 경로인지를 검사한다.

3) BOOL ShowToolbar(BSTR name, BOOL show)

툴바를 보이거나 감춘다.

name : 툴바 이름 (예:그리기, 표, 메모)

4) IXHwpToolbarButton CreateMenuButton(BSTR name, BSTR aid, long style)

메뉴 버튼을 생성한다.

name : 사용자에게 보여질 메뉴 이름

aid : Action ID 문자열, 빈문자열("")을 주면 서브 메뉴를 갖는 메뉴를 생성한다.

style : 현재 사용하지 않음.

5) IXHwpToolbar CreateToolbar(BSTR name, long style)

툴바를 만든다.

name : 툴바 이름

style : 현재 사용하지 않음

6) IXHwpToolbarButton CreateToolbarButton(BSTR name, BSTR aid, long style)

일반 툴바 버튼을 만든다. 생성된 툴바 버튼은 툴바나 메뉴버튼에 추가할 수 있다.

name : 버튼 이름

aid : Action ID 문자열

7) IXHwpToolbar GetToolbar(BSTR name)

존재하는 툴바를 얻어 온다.

8) void ChangeSerializePath(BSTR name)

Automation으로 실행하지 않은 한글의 프레임 상태에 영향을 주지 않도록 Serialize 경로를 변경한다.

나. IXHwpToolbar

1) void InsertToolBarButton(IXHwpToolBarButton ToolBarButton, long pos)

툴바 버튼을 추가한다.

ToolBarButton : IXHwpToolBarLayout.CreateToolBarButton 이나 IXHwpToolBarLayout.CreateToolBarMenuButton으로 생성한 버튼

pos : 추가될 위치 - 0부터 시작하며, 범위를 벗어나면 맨 뒤에 추가 된다.

2) BOOL DeleteToolBarButton(long pos)

툴바 버튼을 삭제한다.

pos : 삭제할 버튼의 위치 - 0부터 시작하며, 범위를 벗어나면 실패한다.

3) IXHwpToolBarButton GetToolBarButton(long pos)

툴바 버튼을 얻어온다.

pos : 툴바 버튼의 위치 - 0부터 시작하며, 범위를 벗어나면 실패한다.

다. IXHwpToolBarButton

1) VARIANT GetHandle()

내부 구현용 - 툴바 버튼의 핸들을 리턴한다.

2) BSTR GetActionName()

Action ID 문자열을 리턴한다.

3) void SetText(BSTR name)

툴바버튼의 이름을 지정한다.

4) BSTR GetText()

툴바버튼의 이름을 리턴한다.

5) long GetStyle()

툴바 버튼의 스타일 각 속성을 OR 연산하여 리턴한다.

00₍₂₎ : 스타일 없음

01₍₂₎ : 문자열 보임

10₍₂₎ : 이미지 보임

11₍₂₎ : 문자열 / 이미지 보임

6) void ModifyStyle(long remove, long Add)

툴바 버튼의 스타일을 수정한다.

remove : 제거할 스타일

Add : 추가할 스타일

예)

toolbarbutton.ModifyStyle(1, 2); // 이미지만 보이도록 한다.

라. IXHwpToolbarMenuButton : IXHwpToolbarButton

IXHwpToolBarButton을 상속받으므로 아래 설명된 함수와 함께 IXHwpToolBarButton의 모든 함수를 사용할 수 있다.

1) void InsertMenuButton(IXHwpToolBarButton ToolBarButton, long pos)

서브 메뉴 버튼을 추가한다.

ToolBarButton : IXHwpToolBarLayout.CreateToolBarButton 이나 IXHwpToolBarLayout.CreateToolBarMenuButton으로 생성한 버튼

pos : 추가될 위치 - 범위를 벗어나면 맨 마지막에 추가된다.

2) BOOL DeleteMenuButton(long pos)

서브 메뉴 버튼을 삭제한다.

pos : 삭제할 버튼의 위치 - 범위를 벗어나면 실패한다.

3) IXHwpToolBarButton GetMenuButton(long pos)

서브 메뉴 버튼을 얻어온다.

pos : 버튼의 위치 - 범위를 벗어나면 실패한다.

※ 참고

한글에서는 툴바 버튼과 메뉴 버튼을 구별하지 않습니다.

즉, `IXHwpToolBarLayout.CreateToolBarButton`을 통해 생성된 버튼은 툴바와 메뉴, 양쪽 모두 사용할 수 있습니다.