PB 的扩展 DLL 开发（超级篇）（二）

(PB 史上第一次开放的开发技术)

上一章里，我们说到 system library 里的函数声明为下面这样的方式。

```
__declspec(dllexport) DWORD __stdcall FuncName(POB_THIS obThis,int nArgCount)
{
    BOOL isnull;
    return 1;
}
```

其中第一个参数 POB_THIS obThis 是每个函数的核心，本篇重点介绍一下核心结构 OB_THIS，它是每个函数的第一个参数。这个结构有点大，通常我们不需要直接访问成员结构，但了解了这个结构的成员，有时候可以很方便地访问一些全局的内容。

```
typedef struct ob_this      {
    void* __vtbl;
    POB_THIS            obthis;             // 兼容性指针
    POB_THIS            rtthis;             //兼容性指针
    PSH_DBG_THIS        dbgthis;            //调试时使用
    ppbstg_anchor       stgthis;            // 指向内存分配的锚，有些函数需要用到它，比如 pbstg_alloc
    UINT                curr_pcode;             // Current pcode Instruction
    BOOL                pcode_done;             // End of pcode block indicator.
    PSHLIST             pBreakPointList;        // 指向断点列表
    INT                 bStep;                  // Processing step mode
    RT_BREAK_PROC       rtBreakCallback[PD_SIZE]; // Debugger call back function
    PVOID               pUserData;              // 给用户使用，存放用户数据。
    PSHLIST             pTransactionList;       // 事务列表
    BOOL                bIgnoreErrors;          // Ignore runtime error flag
    BOOL                bTerminateRuntime;      // Terminate runtime flag
    OB_CLASS_HNDL       clshndl;                // Current class handle
    OB_CLASS_HNDL       appclshndl;             // Application class handle
    OB_EVT_ID           event_id;               // Event ID
    OB_INST_ID          appinst;                // Application 实例对象
    pbstg_subpool       subpool;            //指向内存分配的池，有些函数需要用到它，比如 pbstg_alloc
    RT_MODE             mode;                   // 模式，比如编译状态、EXE 运行状态、调试状态等
    PVOID               smthis;                 // Semantic this pointer
    OB_MODE             obMode;                 // Object manager mode
    UINT                iPCodeCounter;          // PCode execution counter
```

```
    RT_OPT_MODE         opt_mode;           // Optimization mode
    PSHLIST             pDllList;           //已经加入的外部 DLL 列表，即
程序里那些声明 DLL 函数用到的 DLL
    PVOID               curr_pcode_blk;     // Current pcode block
    UINT                iContextCount;      // Number of run contexts
                                            //  only valid in primary
rtthis
    BOOL                bHaltClose;         // Halt close processing flag
    BOOL                bDontTerminate;     // Dont terminate after
error
    UINT                curr_line_block_loc;  // Current line_block loc.
    UINT                last_break_line_no;   // Last line numbers...
    UINT                last_pcode_line_no;   // ...
    PVOID FAR* pThreadStart;            // Points to start of thread
    PVOID FAR* pThreadNode;             // Points to cur. thread node
    UINT                halt_close_nest;    // Count of nested halt close
    PCM_THIS            cmthis;             // Current compiler context

    PSHLIST             pLibraryList;       //库列表，SetLibraryList 函数
就是设置此列表
    PSHLIST             pNameList;          // Library entry name list
    PVOID               pGroupList;         // Pointer to group list nodes
    PVOID               pLookSymKeyFunction;  // Group name hash key
function
    PVOID               scnthis;            // Scanner this pointer
    UINT                iGroupListIncr;     // Group list increment
    UINT                iGroupListPos;      // Group list position
    UINT                iGroupListSize;     // Group list size
    pbstg_subpool       client_subpool;     // Client subpool
    pbstg_subpool       perm_subpool;       // Permanent subpool
    pbstg_subpool       result_subpool;     // Intermediate results
    pbstg_subpool       temp_subpool;       // Temporary subpool
    PSHLIST             group_stack;        // List of groups ids currently in
use
    PVOID               curr_obj_group;
    PVOID               sys_typedef_group;
    PVOID               curr_routnode;
    OB_INST_ID          curr_obinst;    //当前对象实例，比如 json.Set(...)调用
时，Set 函数里的当前对象就指向 json 这个实例。
    PVOID               lib_handle;
    PSHLIST             run_stack;          //  Used to maintain
Runtime Call Stack
    PVOID               rtinst_stack;       // Used to maintain instance list
    LPTSTR              def_appl_groupname; // 创建 target 时指定的
```

application 名称

```
LPTSTR          def_appl_libname; //创建 target 时指定的 pbl ,通常名称与 application 一样，但后缀名是 .pbl
PVOID           appl_group;
BOOL            bInRuntime;
UINT            appl_filter;
PVOID           hExecutable;
UINT            iExeGroupCounter;
PSHLIST         pExeGroupList;          // No longer used
PVOID           pExeResourceHash;       // Resource list for build .EXE
POT_EVAL_NODE       expr_stack;             // maintain params and vars used by pcodes
UINT            expr_stack_size;        // size of expression (data) stack
POT_EVAL_NODE   expr_stack_ptr;         // current stack entry
PSHLIST         arglist_stack;          // NOT USED
PSHLIST         error_list;             // list of errors during application build
BOOL            bGotLinkError;
pbstg_subpool   lvalue_subpool;         // Lvalues
BOOL            bNoDuplicateSymbols;
PSHLIST         unused;                 // UNUSED
OB_CLASS_HNDL   curr_class_hndl;
OB_ARRAY_ID     curr_array_inst;        // Current array inst id.
UINT            ErrorCode;
BOOL            set_return_called;
OB_GROUP_HNDL   sys_group_hndl;
PVOID           pSavedGroupList;        // Pointer to saved group list nodes
PVOID           lmithis;                // Ptr to Lib Mgr anchor
PVOID           curr_runstk_node;       // Ptr to runtime stack node.
ULONG           ps_options;             // PS executable options
OB_EXE_CODE_TYPE exe_code_type;         // CCode or PCode
LPTSTR          pExecutableName;        // current executable
PVOID           evaled_arglist;         // 每个函数调用时，这里是实际参数数组指针，可用 ot_get_....arg(...) 系列函数来取具体参数，也可以 for(int i=0;i<nArgCount;i++){...} 来循环直接取参数
                                        //     from expr_stack and converted as necessary
PVOID           lvalue_info;
UINT            curr_arg_pos; //保留取当前参数的位置。函数开始时，此变量为 0，用 ot_get_....arg(...)每取走一个参数时，此变量会增加。如果取参数的次数大于实际参数个数，会导致程序崩溃。因此取参数时最好要判断此变量不大于参数个数 nArgCount。
POT_EVAL_NODE   return_value;           //
```

```
        POT_EVAL_NODE       called_return_value;    //函数调用结束，实际返回值存放在这里。通常用 ot_set_return_val(...) 来设置，或者 ot_no_ret_val(obThis) 来表明无返回值。

        PVOID               p_error_info;           // Error information for CCode
        LONG                routine_level;          // the routine level for CCode

        ULONG               ulDServFlags;           // Flags for distributed PB
        PVOID               working_group;              // Current working group
        PVOID               callback_data;          // Used to store pointer for
                                                        // callback functions.
        PVOID               callback_data2;             // 2nd instance

        POB_THIS            parentObThis;           // The obthis that was cloned to

        BOOL                    check_for_locked_menu;  // used by 'obinst.cpp' - Currently only used in local object create.
                                                        // can be easily removed


    //************************************************************************
        // file I/O buffers

    //************************************************************************
        LPTSTR              pEntryBuffer;
        TCHAR HUGEPTR* pHugeEntryBuffer;
        LONG                lEntryPos;
        LONG                lEntrySize;


    //************************************************************************
        // Multi-Tasking, Transaction Pooling, Miscellaneous

    //************************************************************************
        PVOID               pTransPool;             // Transaction Pool

        // this growblock is used as an array of groupId's to enable the codegen
        // routines to be thread safe and still be able to load the context.
        PSH_GROWBLOCK       pGroupIdArray;
        PVOID               criticalSection;    // general purpose critical section
```

```
        PVOID           pCmCompilerError;   // so we can decouple compiler
dll

        ULONG           GeneralFlags;
        PVOID           unused_2;           // some free bytes for the
taking ...
        BOOL            bIgnoreLinkErrors;  // ignore link errors.
        UINT            new_next_pcode;     // User set pcode
Instruction
        PVOID           pSharedObjMgr;      // shared object manager
interface

        // trace facility
        LPVOID          traceStruct;        // tracing structure
        BOOL            bTracing;           // global trace enable flag

        // debugger facility
        PSHLIST         pWatchPointList;    // Pointer to watchpoint list
        PSHLIST         dbg_callstk;        // Stack for walking up call stack
        UINT            bStepLine;          // Line number for step to
cursor
        INT             bStepRoutineLevel;  // Depth step command started
at

        // secure runtime session
        ULONG           SecureRuntimeSession; // flags for apps downloaded
over the Web.

        // garbage collection
        DWORD           lGCTimeLimit;       // the min time between garbage
collection runs
        DWORD           lLastGCTime;        // the time of the end of the last
GC run
        BOOL            bGarbageCollecting; // if we are in the process of
garbage collecting

        // object manager request queue
        PVOID           pRequestQueue;      // a pointer to the obj_mgr
request queue
        // pointer to error callback function for handling runtime errors
        PVOID           pErrorCallback;

        // windows dispatch handler
```

```
		PVOID		pWndDispatchHndlr;		// a pointer to the dispatch handler

		// these should all be protected by ob_enter_critical_section()!!
		DWORD		idOwnerThread;		// the handle the of thread that owns the OBTHIS
		BOOL		bActiveSession;		// indicate that this obthis is an active session

		BOOL		bIsAsyncCall;		// indicates if root of call is async

		PVOID		pLocalSession;
		PVOID		local_variables;	// Current local variables to routine
		UINT		num_variables;		// Number of local variables.
		UINT		curr_stack_bias;	// Current stack bias. Used for gc.

		BOOL		bDeferredDelete;	// indicates if we are deferring the delete
							// of an active session until after all requests
							// have completed processing.
		// Context Feature OB_ICONTEXT pointers
		PVOID		pSessOB_ICONTEXT;
		PVOID		pDefltOB_ICONTEXT;

		// Class Definition (a.k.a. metaclass) local session list
		PSHLIST		pSessionList;

		LPTSTR		lpstrTypdefPblName;	// if we want to ignore the default of "pbtypvm70.dll"

		BOOL		bNoMessageBoxForError;	// TRUE if we don't want the message box to come up.
		BOOL		bInSystemError;		// TRUE if we are already executing the system error event.
		PVOID		last_break_routine;	// Last routine a breakpoint was processed in.
		BOOL		bInitDebugMode;		// Are we in the debugger and NOT running from within it.
		PVOID		pContextObject;		// The context object
```

```
        // remote debugger facility
        PVOID           pLocalDebugSession;     // If non-NULL, the local debug
session for Remote debugging
        ULONG           breakpointId;           // unique id for breakpoints

        // GroupLoader stuff.
        PVOID           pGroupLoader;
        //Resulset Marshaling
        PVOID           pResultsetInfo;
        // pointer to error callback state information for functions handling runtime errors
        PVOID           pErrorCallbackState;

        // Exception Handling
        PSHLIST         exception_stack;        // stack containing the checked
exception list
        PVOID           thrown_exception;       // pointer to thrown exception
object,
                                                //    NULL if none pending
        PVOID           curr_exception;         //    pointer to currently
handled exception object,
                                                //    NULL if not currently
handling one
        PSHLIST         gosub_stack;            // stack containing the list of
gosub return locations

        struct ResponseWindowStack
        {
            ResponseWindowStackNode* stack;
            UINT capacity;
            UINT count;
        } response_window_stack;

        PSHLIST         objects_creating;       // A list object being created.
        PVOID           pb_session;             // To be used for PSPP
        PVOID           pMetaObject;            // A pointer to the meta object
created by a func call
                                                // and need to be free in the
next sf_dot call.
        // PB Accessibility service
        /*这部分 pb9 没有
        PVOID           pACCService;
        BOOL            isInOrcaBootStrap;
        LPTSTR          pCurrMissedTypeWhenOrcaBootStrap;
        BOOL            bInNewBuild;
```

```
    PVOID                   pb_sessionA;
    BOOL                     bJITStart;
    BOOL                    bJITOption;
    */
} OB_THIS;
```

OB_THIS 这个结构比较复杂，有必要对它进行一些研究和了解。

如果你对 system library 相关开发方式感兴趣，可到 QQ 群 624409252 共享里大自在的专用目录下下载 DEMO。