مقدمه:

بسیاری از برنامه نویسان Dos معتقدند که ویندوز کار برنامه نویسان را ساده کرده است و در واقع کار مشکلی در پیش ندارند و از زبان به عنوان ابزار استفاده می کنند.در حالی که چنین نیست ; البته در مواقعی که شما Menu ها و Button ها و سایر کنترل ها را به راحتی به فرم اضافه می کنید به نظر می رسد که کار راحتی را انجام داده اید ولی این فقط طراحی ظاهر است نه چیز دیگری.

در واقع ما هر کاری که انجام می دهیم از ویندوز در خواست کرده ایم که این عمل را برای ما انجام دهد پس ما برای اینکه بهتر بتوانیم برنامه بنویسیم و قدر کامل را در دست بگیریم بهتر است با ویندور در مرحله اول آشنا شویم.

قلب ویندوز

تعجب نکنید ، می خواهیم راجع به قلب ویندوز صحبت کنیم یعنی چیزی که اساس سیستم عامل ویندوز است. قلب ویندوز چیزی نیست به جز API (Application Programming Interface) یا رابط برنامه نویسی کاربردی آن ، API شامل بیش از 1000 تابع و پردازه می باشد که برنامه های کاربردی می توانند از آنها استفاده کنند . در حقیقت هر کاری در ویندوز روی می دهد و هر اتفاقی که می افتد با استفاده از این توابع صورت می گیرد.

در حقیقت نوشتن برنامه برای ویندوز از صدا کردن توابع API تشکیل می شود . اگر برنامه نویس ویندوز بگوید که من به توابع API نیازی ندارم در حقیقت او همان استفاده کننده از Tools ابزارهای رایج ویندوز می باشد .

API بخشی از خود ویندوز است ، لذا تمام برنامه های کاربردی ویندوز به همان توابعی دسترسی دارند که برنامه دیگر دسترسی دارد.

در محیط چند و ظیفه ای ویندوز ضروری است که تمام برنامه ها در روش تعریف شده دقیق عمل کنند. با استفاده از توابع API که در تمامی برنامه های کاربردی استفاده می شود این سازگاری به و جود می آید.

شاید شما ندانسته یا ناخواسته تا به حال بارها با API کار کرده باشید. وقتی شما در محیط یک زبان برنامه نویسی مثل Delphi یا Visual Basic یا هر محیط دیگر پروژه جدیدی را آغاز کرده اید. در حقیقت در همین لحظه تابع CreateWindow فراخوانی شده و فرم مربوز در مقابل شما قرار می گیرد که خود این تابع یکی از توابع API است.

API ممکن است پیچیده به نظر برسد ، آن هم با داشتن بیش از 1000 تابع ، ولی API یک امکان واقعا قوی است.

از نظر برنامه نویسی API شامل یک سری زیر برنامه های آماده و کاملا سالم می باشد که در اختیار برنامه نویس قرار می گیرد.
توابع API در کجا هستند؟
حتما این سوال برای شما مطرح شده است که این توابع قدرتمند در کجا قرار دارند ؟ یا اینکه این توابع کجا هستنمد که می توانند به طور اشتراکی در چند برنامه و حتی خود ویندوز مورد ایتفاده قرار گیرند؟ عجله نکنید. ابتدا باید مطالبی در مورد DLL ها بدانیم.

DLL چیست؟

در حقیقت DLL ها هم مانند API ابزاری هستند که شما از آن ها استفاده کرده اید ولی شاید نمی دانستید. اگر نکاهی به دایرکتوری های System،Windows،System۳۲ و یا خود دابرکتوری ویندوز بیاندازید تعداد زیادی از فایل های DLL را می بابید.
Dynamic Linked Library(DLL) : DLL یا کتابخانه پیوندی پویا می باشد که می تواندشامل انواع داده یا کد باشد. درون DLL می توان انواعی از داده ها همانند کد،تصویر ، صوت و... را قرار داد.
نکته جالب فایلهای DLL خاصیت مستقل از زبان برنامه نویسی این فایلها می باشد. یعنی اگر DLL با استفاده Delphi ایجاد شده باشد می توان در Visual Basic یا Visual C++ از آن استفاده کرد.
نکته دیگر DLL ها صرفه جویی در مصرف حافظه می باشد که همین دلیل نام Dynamic را به همراه خود دارند. در واقع هر قسمت از DLL که فراخوانی می شود همان قسمت به حافظه یار می شود و در صورت عدم نیاز از حافظه خارج می شود.
در واقع دلایل استفاده از DLL را می توان موارد زیر نام برد:

1- توانایی اشتراکی کردن کد بین چند برنامه حتی خود ویندوز.
2- استفاده های مجدد از کد های نوشته شده .
3- استفاده بهینه از منابع ویندوز و منابع سیستم.
4- جدا کردن کدهای مختلف از هم.

لازم به ذکر است که DLL ها را می توان با زبان هایی مانند Delphi و یا Visual C++ تولید کرد.
حال مطئنا می توانید جواب ای ن سوال را که API ها در کجا هستند را بدهید. بله API ها در فایلهای DLL ویندوز قرار دارند.
مهمترین DLL ها که بیشترین API ها را در خود جای داده اند عبارتند از Kernel۳۲.DLL وUser۳۲.DLL و Shell۳۲.DLL و... البته DLL های دیگری هم وجود داند.

مقدمه ای درباره پیغام:

شاید تا کنون تحت ویندوز با محیط های چون دلفی ویا #c برنامه نویسی کرده باشید وبا مفهوم رویداد آشنا باشید ولی می دانید که چگونه این روبداد ها را بـه وجـود آورده انـد و مـا چگونـه مـی توانیم این روبداد ها را ایجاد کنیم.آیا تا کنون فکر کرده اید که چگونه می تـوان بـین برنامـه هـای اجرای ارتباط برقرار کرد و حتی به آنها فرمان داد .

پیام چیست؟

هر پیام ، هشداری در مورد برخی از رخدادهاست که از سوی ویندوز برای برنامه ی کاربردی ارسال می شود. به عنوان مثال ، کلیک کردن بر روی دکمه مـاوس، تغییـر انـدازه مجـدد یـک پنجـره یـا فشردن یک کلید بر روی صفحه کلید سبب می شود که ویندوز پیغـامی را بـرای برنامـه کـاربردی ارسال و آن را از آنچه رو ی داده است مطلع سازد.

هر پیام خود را به عنوان یک رکورد رد شده به یک برنامه کاربردی توسط ویندوز احضار می نماید. رکورد مربوز شامل اطلاعاتی در مورد نوع رخداد و اطلاعاتی دیگر در مورد رکورد می باشـد . مـثلا ، شامل مختصات ماوس در زمان فشرده شدن دکمه خواهد بود . نوع رکورد رد شد.

چه مواردی در یک پیغام ویندوز و جود دارد؟

hWnd: دستگبره پنجره ای است که به مقصد آن ارسال می شود.

Message: مقداری ثابت که پبغام را نشان می دهد.

wParam: شامل اطلاعات اضافی پیغام می باشد.

lParam: شامل اطلاعات اضافی پیغام می باشد.

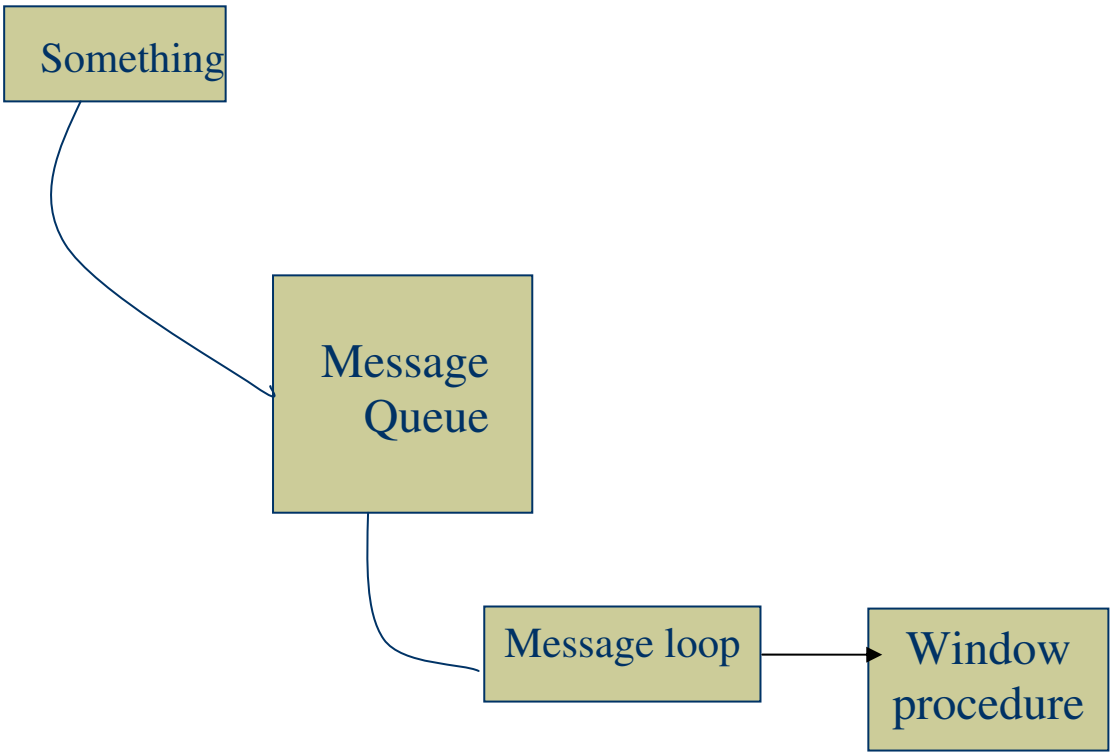
به عنوان مثال وقتی شما کاید چپ موس را فشار می دهید در واقع یک پبغام Wm_LBUTTONDOWN به ویندوز ارسال می کند ; که ویندوز متوجه می شود که شما کلید چپ موس را فشار داده اید.

- چگونه سیستم پیام ویندوز کار می کند؟

هر سیستم پیام کاربردی ویندوز دارای سه مولفه اصلی است:

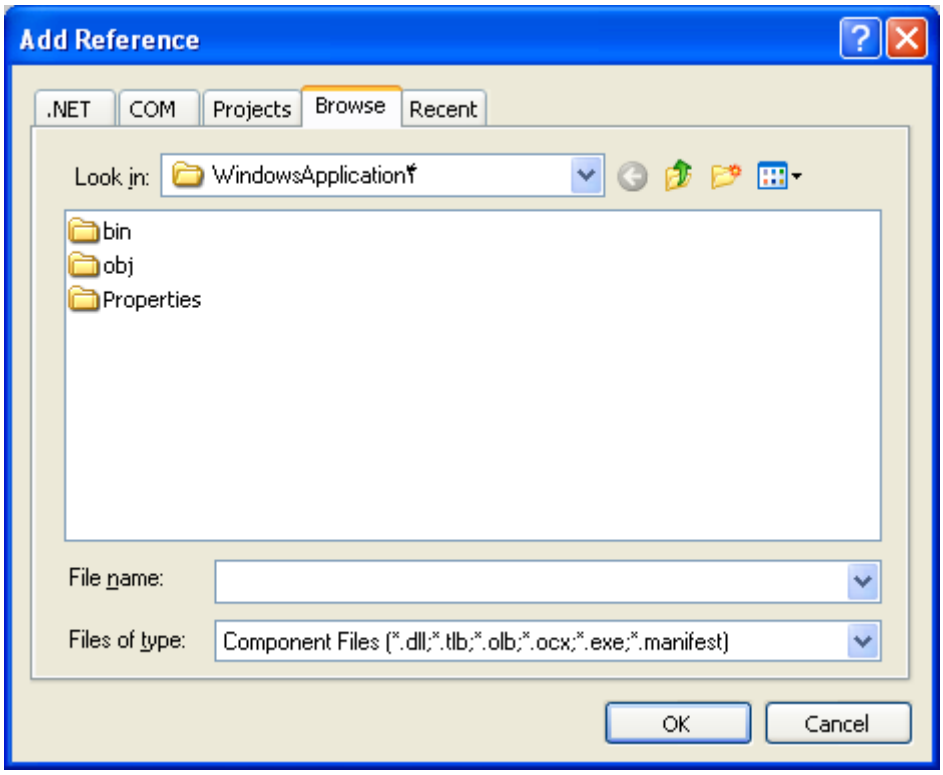
- صف پیام – ویندوز برای هر برنامه کاربردی یک صف پیام تعیین می کند. هر برنامه کاربردی ویندوز باید پیام ها را از این صف گرفته و آنها را به پنجره های مناسب ارسال نماید.
- - حلقه پیام (Message Loop) – منظور سازوکار حلقه ای در یک برنامه ویندوز است که پیام ها را از صف برنامه کاربردی واکشی نموده و آنها را به پنجره مناسب ارسال می نماید و مجددا پیام بعدی را واکشی می کند و آن را به پنجره ی مورد نظر ارسال می کند و الی آخر .

- رویه پنجره – هر پنجره در برنامه کاربردی شما دارای یک رویه پنجره است که پیام ها را دریافت و آن را به حلقه پیام رد می کند. وظیفه ی رویه پنجره این است که هر پیام پنجره را گرفته و متقابلا به آن پاسخ دهد. رویه پنجره نوعی تابع فراخوان متقابل (Callback) است که معمولا پس از پردازش پیام ، یک مقدار (Value) را به ویندوز بر می گرداند.



چگونگی استفاده از توابع API:

اولین قدم برای استفاده از توابع API ،بارگذاری DLL مربوطه در حافظه می باشد.
بارگذاری DLL در حافظه به دو صورت ایستا و پویا صورت می گیرد.
الف)حالت ایستا:در این حالت DLL مربوطه از اول اجرای برنامه تا پایان اجرا در حافظه باقی می ماند ومی توان از توابع داخل آن استفاده کرد.این روش حافظه زیادی از سیستم را اشغال می کند.برای بارگذاری به صورت ایستا ابتدا در منوی Project گزینه Add Reference را انتخاب کرده تا پنجره زیر باز شود در این صورت می توان در قسمت Browse فایل DLL مورد نظر را انتخاب کرده تا به پروژه اضافه شود.



ب)حالت پویا:در این حالت DLL مروبطه فقط برای استفاده خاص و در زمان مورد نیاز در حافظه قرار می گیرد و سپس بلافاصله از حافظه خارج می شود.برای این منظور باید از روش زیر استفاده کرد:
ابتدا باید `using System.Runtime.InteropServices;` را فراخوانی کرد و در مرحله بعد با استفاده از دستور `[DllImport("DLL Name")]` DLL مورد نظر را در حافظه بار گذاری کرد.
مثال: `[DllImport("user۳۲.dll")]`

بعد از بارگذاری DLL مربوطه باید خود تابع API را در برنامه تعریف کرد که این عمل به صورت زیر انجام می گیرد:
برای مثال تابع زیر برای ارسال پیغام به سیستم عامل استفاده می شود.این تابع کاربرد های زیادی دارد.

نکته: برای بدست آوردن لیست توابع به همراه پارامتر های آن می توانید به فایلAPI Viewer که به همراه VB۶ نصب می شود استفاده کنید.

```
public static extern int SendMessage(IntPtr hWnd,
int Msg, int wParam, int lParam);
```

بعد از این مرحله به راحتی می توانید این تابع را صدا زده و از آن استفاده کنید.

برای استفاده از پیغام ها ابتدا باید ثابت آن را به صورت زیر تعریف کرد.

```
public const int Message Name = Const Value;
```

مـثـال:

```
public const int WM_SYSCOMMAND = ۰x۰۱۱۲;
```

حال برنامه زیر معجزه می کند و شما به راحتی می توانید فرم با هر شی دیگری را جابه جا کنید.

```
public const int WM_SYSCOMMAND = ۰x۰۱۱۲;
[DllImport("user۳۲.dll")]
public static extern int SendMessage(IntPtr hWnd,
int Msg, int wParam, int lParam);
[DllImport("user۳۲.dll")]
public static extern bool ReleaseCapture();

private void Form۱_MouseDown(object sender,
MouseEventArgs e)
  {
   ReleaseCapture();
   SendMessage(Handle, WM_SYSCOMMAND, ۰Xf۰۱۲, ۰);
  }
```

نکته:برای حرکت Button۱فقط کافیست بنویسید Button۱.Handle

مثال:این برنامه نوع زبان صفحه کلید را تغییر می دهد:

```
[DllImport("user۳۲")]
public static extern int
ActivateKeyboardLayout(int HKL, int flags);

private void button۱_Click(object    sender,
   EventArgs e)
    {
        فارس می شود
        ActivateKeyboardLayout(۰x۰۰۰۰۰۴۲۹,۱);
        انگلیسی می شود
         ActivateKeyboardLayout(۰x۰۰۰۰۰۴۰۹,۱);
     }
```

مثال:بستن برنامه Windows Media Player با استفاده از برنامه زیر انجام می گیرد.
در ابتدا با استفاده از تابع APIی FindWindow، Handle برنامه مورد نظر را بدست آورده وبا ارسال پیغام Wm_Close برنامه مورد نظر بسته می شود.

```
[DllImport("user۳۲.dll", CharSet = CharSet.Auto,
SetLastError = true)]
        private static extern int FindWindow
(String  lpClassName , string lpWindowName) ;
        [DllImport("user۳۲.dll", CharSet =
CharSet.Auto, SetLastError = true)]
        private static extern int SendMessage(int
hwnd, int Msg, IntPtr wParam, IntPtr lParam);
        public const int WM_CLOSE = ۰x۱۰;
        int hwnd;
            hwnd = FindWindow(null , "Windows
Media Player");
            hwnd = FindWindow("WMPlayerApp",
"Windows Media Player");
            SendMessage(hwnd, WM_CLOSE,
IntPtr.Zero ,IntPtr.Zero );
```

نکته : برای بدست آوردن Handle,ClassName,Caption هر برنامه می توانید از نرم افزار SPY++ که همراه را با VB۶ نصب می شود استفاده کنید.

گرفتن کلیه پیغامهای رسیده به پنجره:

با تعریف این تایع کلیه پیغام های رسیده به فرم از این تابع عبور می کنند که با صدا زدن دستور base.WndProc(ref message); کلیه آنها را به کلاس بالاتر ارسال می کنیم.

برنامه زیر اگر بر روی کلاس BUTTON۱ کلیک شد پیغام نشان می دهد.
با استفاده از این روش می توانیم انجام بعضی از اعمال را از یکسری اشیاء بگیریم.

```
protected override void WndProc(ref Message
message)
 {
     if (message.HWnd == BUTTON1.Handle)
         {
           if (message.Msg == WM_LBUTTONDOWN)
                    MessageBox.Show("");
          }

                base.WndProc(ref message);
 }
```