

برنامه نویسی شبکه و اینترنت در VB بخش اول

مروری بر TCP/IP

نکته : مطالب زیر تنها در حد یک یادآوری می باشد . اگر اطلاعات کمی در مورد TCP/IP دارید به کتابهای موجود مراجعه کنید .

پروتکل Protocol : قراردادی است برای برقراری ارتباط در شبکه

مدل TCP/IP : مجموعه ای از پروتکل‌های ارتباطی مرتبط بهم است که مکانیزمها و سرویسهای مورد نیاز جهت برقراری ارتباط در اینترنت را مهیا می کنند . این مدل شامل ۴ لایه است :

۱ - لایه کاربرد Application Layer : شامل برنامه های کاربردی و پروتکل‌هایی مثل Http ، Ftp ، Smtip ، Pop و Telnet می باشد .

۲ - لایه انتقال Transport Layer : این لایه شامل دو پروتکل TCP و UDP است . پروتکل TCP وظیفه کنترل رسیدن بسته های داده به مقصد (TCP/IP داده ها را به بسته های کوچکی تقسیم می کند که هر بسته حاوی آدرس فرستنده ، گیرنده و شماره بسته می باشد) ، تصحیح خطا و مرتب سازی بسته ها را برعهده دارد . UDP پروتکلی شبیه TCP است با این تفاوت که هیچ ضمانتی برای رسیدن بسته های اطلاعاتی در آن وجود ندارد و معمولاً در انتقال صوت و ویدئو روی اینترنت استفاده می شود .

۳ - لایه اینترنت Layer Internet : شامل پروتکل IP است که مسئول مسیریابی بسته های اطلاعاتی می باشد .

۴ - لایه دسترسی به شبکه Link Layer : شامل بخشی از هسته سیستم عامل و نیز درایورهای واسط شبکه برای کار با سخت افزار شبکه می باشد .

سوکت Socket و پورت Port : سوکت یک ورودی انتزاعی در لایه انتقال می باشد که برای ایجاد ارتباطات مختلف TCP/IP بکار می رود . اغلب برنامه های کاربردی که از TCP و UDP استفاده می کنند ، عملیات انتقال اطلاعات خود را با ساخت یک سوکت و سپس انجام یکسری عملیات روی آن انجام می دهند . این عملیات عبارتند از :

۱ - عملیات کنترلی : شامل اختصاص یک شماره پورت به سوکت ، initiate کردن یا accpet کردن یک ارتباط ، از بین بردن سوکت

۲ - عملیات انتقال داده : شامل نوشتن داده روی سوکت و خواندن داده از سوکت

۳ - عملیات بررسی وضعیت : مثل پیدا کردن آدرس IP مربوط به سوکت ، پیدا کردن شماره پورت سوکت و غیره

HTTP : پروتکل انتقال داده برای وب است .

FTP : پروتکل انتقال فایل روی اینترنت است .

SMTP و POP : پروتکل‌های ارسال و دریافت email می باشند .

کنترل Web Browser – ساخت مرورگر صفحات وب

برنامه Internet Explorer یا iexplore.exe در واقع برنامه کوچکی است که وظیفه اصلی آن ایجاد چارچوبی برای بهم پیوستن عناصر مختلف است و این عناصر هستند که وظایف اصلی مثل load کردن صفحات وب ، اجرای کدهای Html و غیره را انجام می دهند . اصلی ترین عنصری که مستقیماً توسط iexplore.exe استفاده می شود کنترل Webbrowser (موجود در فایل shdocrw.dll) می باشد . وظیفه این فایل dll ، عبارت است از حرکت بین صفحات وب ، مدیریت تاریخچه صفحات دیده شده و غیره . این فایل خود از فایل دیگری بنام Mshtml.dll استفاده می کند که وظیفه آن بررسی و اجرای فایل های html است . مایکروسافت به برنامه نویسان این امکان را داده که بتوانند در برنامه هایشان از کنترل webbrowser استفاده کنند . با استفاده از این کنترل می توان به سادگی یک مرورگر وب تقریباً کامل ساخت .

خصوصیات کنترل Webbrowser :

- Webbrowser علاوه بر خواص استاندارد مثل width، height و ... خواص زیر را دارد :
- 1 – Busy : اگر در حال load کردن یک صفحه یا در حال جستجو در وب باشد این خاصیت True است . توسط متد Stop می توان عملیات جاری را متوقف کرد .
 - 2 – Container : ارجاع به شی نگهدارنده کنترل webbrowser
 - 3 – Document : ارجاع به صفحه html فعلی . برای کار با این صفحه html می توان از خواص و متدهایی شی Document استفاده کرد .
 - 4 – LocationName : حاوی آدرس محلی است که اکنون در کنترل webbrowser، load شده است . اگر این محل یک صفحه html باشد عنوان آن صفحه خواهد بود و اگر این محل یک فایل در شبکه باشد مسیر کامل آن فایل خواهد بود .
 - 5 – LocationURL : حاوی url محلی است که فعلاً در کنترل webbrowser، load شده است .
 - 6 – Offline : اگر کنترل webbrowser در حالت عدم اتصال باشد مقدار آن True و در غیراینصورت False است .
 - 7 – Parent : فرمی را نشان می دهد که کنترل webbrowser در آن قرار دارد .
 - 8 – ReadyState : وضعیت کنترل webbrowser را برمی گرداند .

متدهای کنترل webbrowser : این متدها مربوط به مرور در صفحات وب هستند :

- 1- GoBack : در لیست تاریخچه url ها ، یکی به عقب برمی گردد .
- 2 – GoForward : در لیست تاریخچه url ها ، یکی به جلو می رود .
- 3 – GoHome : به homepage مرورگر می رود .
- 4 – Navigate : به یک url یا فایل می رود . ساختار این متد بصورت زیر است :

URL Navigate [x][Headers][PostData][TargetFrameName][Flags],

URL آدرس مقصد می باشد . Flags نحوه باز شدن آدرس مقصد را تعیین می کند . اگر این پارامتر ذکر نشود آدرس جدید در پنجره فعلی باز خواهد شد و به لیست تاریخچه اضافه شده و اگر کپی آن در cache temporary موجود باشد از آنجا خوانده می شود . مقادیر پارامتر Flags عبارتند از :

- NavOpenInNewWindow : آدرس جدید را در پنجره جدیدی باز می کند .
- NavNoHistory : به لیست تاریخچه اضافه نمی شود بلکه جایگزین صفحه فعلی می شود .
- NavNoReadFromCache : صفحه جدید از cache خوانده نمی شود .
- NavNoWriteToCache : صفحه جدید روی cache نوشته نمی شود

Event های کنترل webbrowser : این event ها مربوط به مرور در وب و تغییر حالت آن هستند :

- 1 – CommandStateChange : برای فعال یا غیرفعال کردن دکمه های Forward و Back در مرورگر استفاده می شود . شکل کلی فراخوانی این event بصورت زیر است :

(Command As Long, ByVal Enable As Boolean Private Sub WebBrowser1_CommandStateChange(ByVal

که command فرمانی است که حالت فعال آن تغییر کرده است و دو مقدار می گیرد : 1 و 3 که به ترتیب معادل فرمانهای GoForward و GoBack هستند .

Enable فعال یا غیرفعال بودن فرمان را تعیین می کند .

2 – DocumentComplete : این event زمانی فعال می شود که صفحه در حال load شدن به حالت

ReadyState_Complete برود . شکل کلی فراخوانی این event بصورت زیر است :

(Variant Private Sub WebBrowser1_DocumentComplete(ByVal pDisp As Object, URL As

که pDisp ارجاعی به کنترل webbrowser است که event در آن رخ داده است و URL آدرس صفحه در حال load شدن است .

3 – DownloadBegin : این event در آغاز حرکت به صفحه جدید روی می دهد و هیچ پارامتری نمی گیرد . مرورگر می تواند در این event پیغامی برای شروع عملیات جدید نشان می دهد .

4 – DownloadComplete : این event در پایان عملیات یا در صورت انصراف کاربر یا بروز خطا روی می دهد .

5 – ProgressChange : با بروز هر تغییری در وضعیت load ، این event روی می دهد . شکل کلی فراخوانی آن بصورت زیر است :

(Progress As Long, ByVal ProgressMax As Long Private Sub WebBrowser1_ProgressChange(ByVal

که Progress نشان دهنده پیشرفت عملیات (بایتهای load شده) است . پارامتر ProgressMax تعداد کل بایتهایی که باید load شوند را نشان می دهد بنابر این :

$$\text{load} = (\text{Progress} / \text{ProgressMax}) * 100$$
 درصد پیشرفت عملیات load

یک مثال ساده :

از منوی project مورد components را انتخاب کنید و از لیست کنترلها ، Microsoft Internet Controls را به toolbar خود اضافه کنید . یک کنترل WebBrowser روی فرم قرار دهید و سائز آنرا به اندازه ابعاد فرم خود قرار دهید . یک textbox و یک دکمه روی فرم قرار دهید . کد زیر را برای event مربوط به کلیک دکمه بنویسید :

WebBrowser.Navigate textbox.text

کنترل Internet Transfer - قسمت اول

مقدمه : کنترل Internet Transfer نسبت به کنترل WebBrowser که در روزهای قبلی معرفی شد در سطح پایینتری قرار دارد . این کنترل با استفاده از دو پروتکل HTTP و FTP می تواند داده ها را منتقل کند . این کنترل زمانیکه از پروتکل HTTP استفاده می کند با همان روش کنترل WebBrowser به سرویس دهنده صفحات وب متصل می شود اما بجای آنکه صفحه وب را نمایش دهد متن Html صفحه را بازبایی می کند . همچنین زمانیکه این کنترل از پروتکل FTP استفاده می کند قادرست فایلها را بین کامپیوترهای روی شبکه منتقل سازد .

اتصالات HTTP : همانطور که می دانید ، پروتکل HTTP استاندارد وب می باشد . صفحات وب با زبان Html نوشته می

شوند و انتقال آنها از server به client توسط پروتکل HTTP صورت می گیرد .
متد OpenURL: ساده ترین راه استفاده از کنترل IT متد OpenURL است . شکل کلی این متد بصورت زیر است :

Inet.OpenURL(url,DataType)x

که url آدرس صفحه وب و DataType نوع داده بازیابی شونده است و دو مقدار icString (داده متنی) یا icByteArray (داده باینری) را می گیرد . مقدار بازگشتی این متد ، داده های منتقل شده است .
این متد بصورت سنکرون کار می کند یعنی در تمام مدت کار آن برنامه نمی تواند کار دیگری انجام دهد . اگر از icByteArray استفاده کنید باید مقدار بازگشتی آنرا در یک آرایه بایت قرار دهید .
مثال 1 : از بخش Component در منوی Project مورد Microsoft Internet Transfer Control 6.0 را به toolbar خود اضافه کنید . سپس یک کنترل IT روی فرم قرار دهید و همچنین یک Rich Textbox و یک دکمه روی فرم قرار دهید و کد زیر را برای event مربوط به کلیک دکمه بنویسید :

TextBox.text=Inet.OpenURL("www.microsoft.com",icString)x

مثال 2 : کد زیر داده های باینری را از اینترنت خوانده و آنها را در یک فایل ذخیره می کند :

```
byte Dim b() as  
B(=Inet.OpenURL(ftp://ftp.microsoft.com/test.zip,icByteArray)x  
test.zip" For Access Write As #1" & App.path Open  
Put #1,b()x  
1# Close
```

رویداد StateChanged : کنترل IT فقط یک event دارد که StateChanged می باشد . این event زمانی روی می دهد که State کنترل تغییر کند . State هر اتفاقی است که برنامه باید از آن مطلع شود . تعریف کلی این event بصورت زیر است :

Inet_StateChanged(ByVal NewState As Integer)x

که NewState مقداری است که حالت جدید را بیان می کند . مقادیر ممکن این پارامتر عبارتند از :

- icNone : حالت تغییر نکرده است .
- icResolvingHost : در حال جستجوی آدرس IP کامپیوتر موردنظر .
- icHostResolved : آدرس IP کامپیوتر موردنظر یافت شد .
- icConnecting : در حال اتصال به کامپیوتر مقصد
- icConnected : اتصال به کامپیوتر مقصد برقرار شد .
- icRequesting : در حال ارسال درخواست به کامپیوتر مقصد
- icRequestSent : درخواست به کامپیوتر مقصد ارسال شد .
- icReceivingResponse : در حال دریافت پاسخ از کامپیوتر مقصد .
- icResponseReceived : پاسخ کامپیوتر مقصد دریافت شد .
- icDisconnecting : در حال قطع اتصال با کامپیوتر مقصد .
- icDisconnected : اتصال مقصد با موفقیت قطع شد .
- icError : در ارتباط با کامپیوتر مقصد خطایی رخ داده است .
- icResponseCompleted : تکمیل پاسخ - تمام داده ها دریافت شد .

تشخیص خطا در عملیات انتقال داده اهمیت بالایی دارد و StateChanged در صورت بروز هر خطایی مقدار icError را برمی گرداند و اطلاعات خطا را در دو خاصیت ResponseCode و ResponseInfo برمی گرداند .

انتقال داده بصورت آسنکرون : کنترل IT متدهای انعطاف پذیر دیگری هم دارد که آسنکرون هستند و اجازه می دهند تا همزمان با عملیات انتقال داده ، برنامه به وظایف دیگری هم بپردازد . این متدها با استفاده از Event Driven Model کار می کنند . بدین معنی که وقتی برنامه درخواست انتقال داده ای را می دهد کنترل IT درخواست را در زمینه برنامه انجام

مي دهد و برنامه ازاد است تا به كارهاي ديگرش پردازد . زمانيكه داده ها بازيايي شود ، داده ها را از بافر داخلي كنترل IT مي خواند .

متد GetChunk : در عمليات انتقال آسنكرون ، بايستي داده را توسط اين متد از بافر داخلي كنترل IT بگيريم :

Inet.GetChunk(datasize[,datatype])x

كه پارامتر datasize از نوع long بوده و تعيين مي كند چند بايت از بافر خوانده شود و پارامتر اختياري datatype نوع داده را مشخص مي كند و مي تواند مقادير icString و icByteArray را بگيرد .
زمانيكه StateChanged وارد حالتهاي icResponseReceived و يا icResponseCompleted شد بايد از GetChunk استفاده كنيد . بدين صورت كه از يك حلقه استفاده مي كنيم تا كل بافر را بخوانيم :

```
Integer)x Private Sub Inet_StateChanged(Byval State as  
Dim temp1,temp2  
Select Case State  
icResponseCompleted Case  
temp1=""x  
temp2=""x  
Do  
temp1=Inet.GetChunk(512,icString)x  
temp1 & temp2=temp2  
Loop Until temp1=""x  
End Select  
End Sub
```

براي بالابردن كارايي ، بهتر است از قطعات كوچك (بين 512 تا 1024 بايتي) استفاده كنيد .

متد Execute : و اما انعطاف پذيرترين متد كنترل IT ، متد Execute است . فرمت كلي اين متد بصورت زير است :

Inet.Execute(url,Command,Data,RequestHeaders)x

كه url آدرس مقصد ، Command فرماني است كه به كامپيوتر مقصد داده مي شود و Data و RequestHeaders اطلاعات اضافي لازم براي اجراي فرمان داده شده است . فرمانهاي Command همان فرمانهاي HTTP هستند كه عبارتند از :

- GET : دريافت داده ها از كامپيوتر مقصد

- HEAD : دريافت اطلاعات header از كامپيوتر مقصد

- POST : ارسال اطلاعات لازم براي تكميل درخواست

- PUT : ارسال فايل براي كامپيوتر ميزبان (upload)

فرمان GET پركاربردترين فرمان متد Execute است و داده هاي خوانده شده را در بافر داخلي بافر كنترل IT قرار مي دهد تا بتوان با متد GetChunk آنها را بازيايي نمود .

مثال :

Inet.Execute http://www.microsoft.com,"GET"x

ساير خواص كنترل IT :

- AccessType : نوع دسترسي كنترل IT به اينترنت را مشخص مي كند و سه مقدار مي تواند بگيرد :

icUseDefault : استفاده از تنظيمات رجيستري براي دسترسي به اينترنت

icDirect : اتصال مستقيم كنترل IT به اينترنت

icNamedProxy : اتصال به اينترنت توسط پروكسي

- Document : نام صفحه پيش فرض كه در متد Execute از آن استفاده مي شود . اگر به متد Execute پارامتر url را ندهيد از اين صفحه پيش فرض استفاده مي كند .

- Password : کلمه رمز عبور کامپیوتر میزبان FTP
 - Procolot : نوع پروتکل مورد استفاده در متد Execute را مشخص می کند و 5 مقدار می تواند بگیرد :
 icUnknown : نامعلوم
 icDefault : پروتکل پیش فرض
 icFTP : پروتکل FTP
 icHTTP : پروتکل HTTP
 icHTTP : پروتکل حفاظت شده HTTP
 - Proxy : نام میزبان پروکسی
 - RequestTimeout : مدت زمانی که کنترل IT صبر می کند تا اطلاعات را دریافت کند . اگر این خاصیت صفر باشد کنترل تا هر زمان که لازم باشد برای دریافت پاسخ صبر می کند . در حالت سنکرون (متد OpenURL) بعد از سپری شدن این مدت زمان ، یک خطا تولید می شود و در حالت آسنکرون (متد Execute) رویداد StateChanged مقدار خطا را بر می گرداند
 - ResponseCode : بعد از بروز حالت icError این خاصیت کد خطا را می دهد .
 - ResponseInfo : توضیحی درباره خطا
 - StillExecuting : اگر True باشد یعنی کنترل مشغول انجام کار است .
 - URL : آدرس مقصد در متدهای OpenURL و یا Execute
 - UserName : نام کاربر برای ورود به کامپیوتر میزبان FTP

کنترل Internet Transfer - قسمت دوم

اتصالات FTP

پروتکل FTP علاوه بر نقل و انتقال فایل بین دو کامپیوتر ، می تواند نوعی مدیریت فایل (مثل حذف فایل یا ایجاد پوشه) روی کامپیوتر مقصد را انجام دهد . FTP در انتقال فایل بسیار قویتر از HTTP است ولی به مراتب پیچیده تر از HTTP می باشد اما کنترل IT این پیچیدگیها را از دید برنامه نویس مخفی کرده است .
 برای کار با سرورهای FTP باید به آنها Login نمود . نوع خاصی از Login به نام Anonymous Login (ورود ناشناس) وجود دارد که با آن کاربران می توانند بدون محدودیت از سایت FTP استفاده کنند . توجه کنید که حتی برای ورود ناشناس هم نیاز به نام کاربر و کلمه عبور است . برای ارسال نام کاربر و کلمه عبور از خواص username و password کنترل IT استفاده می شود . اگر خاصیت username خالی باشد (blank) ، کنترل IT بطور خودکار از anonymous استفاده می کند و آدرس email کاربر بعنوان passowrd استفاده می شود .
استفاده از متد OpenURL : متد OpenURL ساده ترین راه انجام عملیات FTP است . دستور زیر از یک سایت FTP لیست می گیرد :

```
Text.text=Inet.OpenURL("ftp://ftp.microsoft.com",icString)x
```

برای خواندن فایل از یک سایت FTP باید در حالت باینری کار کرد :

```
b()=Inet.OpenURL("ftp://ftp.microsft.com/test.zip",icByteArray)x
```

استفاده از متد Execute : متد Execute قابلیتهای بیشتری دارد و اجرای آن در FTP نیاز به دو پارامتر دارد :

```
Inet.Execute(url,operation)x
```

که url آدرس سایت FTP به همراه نام و مسیر فایل و پارامتر operation یک فرمان FTP است . کنترل IT با داده های خوانده شده FTP به دو طریق رفتار می کند :

برخی از داده ها مثل پاسخ فرمان DIR در بافر کنترل IT قرار می گیرد و باید آنها را با متد GetChunk خواند .

برخی دیگر از داده ها مثل فایل خوانده شده با فرمان GET مستقیماً روی دیسک نوشته می شوند و دیگر نیازی به استفاده از متد GetChunk نیست .

فرامین FTP بسیار قوی هستند و حتی به شما این امکان را می دهند که فایلها را به روی کامپیوتر مقصد کپی کنید ، به پوشه های کامپیوتر مقصد بروید ، فایلها را حذف کنید و یا تغییر نام دهید . البته باید توجه داشت که فرامین قابل اجرا به نوع ورود به سیستم FTP بستگی دارد . اگر با کاربر anonymous به یک سایت FTP وارد شوید تنها می تواند فایلها را ببیند و آنها را download کنید .

مهمترین فرامین FTP عبارتند از :

- CD path : به دایرکتوری path می روید .
- CDUP : به یک دایرکتوری بالاتر می رود .
- CLOSE : بستن اتصال FTP
- DELETE file1 : حذف فایل file1
- DIR file1 : جستجوی فایل file1 روی دایرکتوری جاری
- MKDIR path : ایجاد یک دایرکتوری با نام path
- PUT file1 file2 : فایل file1 را از کامپیوتر مبدا روی فایل file2 در کامپیوتر مقصد کپی می کند .
- PWD : نام دایرکتوری جاری در کامپیوتر مقصد را برمی گرداند .
- QUIT : قطع اتصال FTP
- GET file1 file2 : فایل file1 را از کامپیوتر مقصد روی فایل file2 در کامپیوتر مبدا کپی می کند .
- file2 RENAME file1 : تغییر نام فایل file1 به file2
- RMDIR path : حذف دایرکتوری path در کامپیوتر مقصد
- SIZE file1 : بدست آوردن تعداد بایت های فایل یا دایرکتوری file1

مثال :

```
test.zip")x Inet.Execute("ftp://ftp.microsoft.com","GET
```

کنترل WinSock - قسمت اول

مقدمه :

کنترل WinSock نسبت به تمام کنترل‌های اینترنت در سطح پاینتری قرار دارد . این کنترل امکان ایجاد سرویس‌های شبکه ای مبتنی بر پروتکل‌های TCP و UDP را مهیا می‌کند . عبارت دیگر توسط این کنترل می‌توان برنامه‌های کاربردی Client/Server (سرویس گیرنده / سرویس دهنده) ایجاد و با استفاده از پروتکل TCP و یا UDP بین آنها ارتباط برقرار نمود .

با تنظیم خصوصیات و فراخوانی متدهای این کنترل می‌توانید به راحتی به یک کامپیوتر راه دور متصل شوید و داده‌ها را در هر دو جهت جابجا نمائید . نمونه کاربرهایی که می‌توان با این کنترل ایجاد نمود : Client-server chat ، Mail client ، Mail server ، Proxy Server ، Game Network ، Port Scanner ، پیاده‌سازی الگوریتم‌های موازی و ...

مبانی TCP :

پروتکل کنترل اینترنت (Transfer Control Protocol) اجازه می‌دهد یک اتصال (Connection) را از طریق سوکت (socket) به یک کامپیوتر راه دور (Computer Remote) ساخته و استفاده کنید . با استفاده از این اتصال ، هر دو کامپیوتر می‌توانند داده‌ها را بین خودشان انتقال دهند . برقراری ارتباط از طریق TCP همانند صحبت کردن با تلفن است که باید حتماً اتصالی بین دو کامپیوتر صورت گیرد تا بتوانند با هم ارتباط برقرار کنند . اگر یک برنامه Client می‌سازید بایستی بدانید که نام یا آدرس IP کامپیوتر Server چیست (Remote Host IP) و همچنین از طریق چه پورتی می‌توانید به آن متصل شوید (Remote Port) . حال بایستی به آن پورت Connect کنید . همچنین اگر یک برنامه Server می‌سازید بایستی پورتی را که روی آن به درخواست‌ها گوش می‌دهید مشخص کنید (LocalPort) و سپس به پورت گوش دهید (Listen) . زمانیکه یک کامپیوتر Client تقاضای یک اتصال را می‌دهد Server این درخواست را Accept می‌کند . زمانیکه یک اتصال ساخته می‌شود ، هر دو کامپیوتر می‌توانند داده را فرستاده و دریافت کنند .

مبانی UDP :

پروتکل دیتاگرام کاربر (User Datagram Protocol) پروتکلی بدون اتصال (Connectionless) است . برخلاف TCP ، کامپیوترها نیاز به برپا کردن یک اتصال ندارند بنابراین یک برنامه می‌تواند یک client و یا یک server باشد . برقراری ارتباط در UDP شبیه ارسال نامه از طریق پست است . برای انتقال داده توسط UDP ابتدا باید Local Port کامپیوتر Client تنظیم گردد . کامپیوتر Server تنها بایستی RemoteHost را برابر آدرس کامپیوتر Client قرار دهد و همچنین Remote Port را همان Local Port کامپیوتر Client قرار دهد . سپس دو کامپیوتر می‌توانند داده‌ها را بین خود جابجا کنند .

استفاده از کنترل WinSock :

- 1 - انتخاب پروتکل: در زمان استفاده از کنترل WinSock اولین کاری که باید انجام دهید انتخاب یکی از پروتکل‌های TCP یا UDP است . طبیعت برنامه‌ای که شما می‌سازید نوع پروتکلی را که باید استفاده کنید مشخص می‌کند . چند سوال زیر به شما کمک می‌کند که پروتکل مورد نیازتان را انتخاب کنید :
 - آیا برنامه شما در زمانیکه داده فرستاده می‌شود یا دریافت می‌شود نیاز به اطلاعاتی از طرف Server یا Client دارد ؟ اگر چنین است بایستی یک اتصال TCP قبل از ارسال یا دریافت داده ایجاد شود .
 - آیا داده بسیار بزرگ است (مثل تصویر یا فایل‌های صوتی) ؟ زمانیکه یک اتصال TCP ساخته می‌شود پروتکل TCP اتصال را باقی نگه می‌دارد و درستی ارسال داده تضمین شده است . این اتصال در هر حال به منابع محاسباتی بیشتری نیاز دارد و بنابراین پرهزینه تر است .

- آیا داده متناوب ارسال می شود یا در یک نشست (Session) ارسال خواهد شد ؟ برای مثال اگر شما یک برنامه می سازید که کامپیترهای مشخصی را در یک زمان خاص از انجام شدن عملیاتی مطلع می کند پروتکل UDP مناسب تر است . پروتکل UDP همچنین برای ارسال مقادیر کوچک داده ای مناسب تر می باشد .

2 - تنظیم پروتکل : برای تنظیم پروتکلی که می خواهید در برنامه تان از آن استفاده کنید در زمان طراحی برنامه خاصیت Protocol کنترل WinSock را برابر sckTCPProtocol و یا sckUDPProtocol قرار دهید . همچنین می توانید پروتکل خود را توسط کد زیر تنظیم کنید :

```
WinSock.Protocol=sckTCPProtocol
```

3 - مشخص کردن نام کامپیوتان : برای اتصال به کامپیوتر راه دور بایستی آدرس IP و یا نام کامپیوتر را بدانید . نام کامپیوتر در Control Panel/Network/Identification موجود است . در صورتیکه می خواهید دو برنامه Client و Server خود را روی یک کامپیوتر تست کنید از آدرس IP 127.0.0.1 برای هر دو استفاده کنید اما اگر دو برنامه را روی دو کامپیوتر مجزا در شبکه قرار داده اید با اجرای دستور ipconfig در DOS Prompt می توانید آدرس IP کامپیوتر ها را بدست آورید .

4 - ایجاد اتصال TCP : در زمان ساخت برنامه ای که از پروتکل TCP استفاده می کند ابتدا باید تصمیم بگیرید که این برنامه Client است یا Server . برای ساخت یک برنامه Server بایستی روی یک پورت خاص Listen کنید . زمانی که Client تقاضای یک اتصال را می دهد ، برنامه Server می تواند آنرا Accept کند و بنابراین اتصال کامل شده است . حال Client و Server می توانند با هم ارتباط داشته باشند .

مراحل زیر ساخت یک سرور چت ساده بر مبنای TCP را نشان می دهد :

- از منوی Project گزینه Components را انتخاب کنید و در لیست Component ها مورد Microsoft WinSock 6.0 را انتخاب کنید .

- یک کنترل WinSock در فرم خود قرار دهید و نام آنرا tcpserver بگذارید

- دو textbox با نامهای txtSendData و txtReceiveData و نیز یک دکمه در فرم قرار دهید .

- کد زیر را در رویداد Form_Load بنویسید :

```
Tcpserver.LocalPort=1000
tcpserver.Listen
```

- زمانی که درخواستی از طرف Client می آید رویداد ConnectionRequest اجرا می شود . در این رویداد ابتدا باید چک کنید که حالت کنترل بسته باشد . اگر چنین نیست اتصال را قبل از پذیرفتن اتصال جدید ببندید . سپس تقاضا را بر اساس پارامتر requestID می پذیریم :

```
(requestID As Long Private Sub tcpserver_ConnectionRequest(ByVal
tcpserver.Close sckClosed Then <> If tcpserver.State
tcpserver.Accept requestID
End Sub
```

- حال اتصال بین Client و Server برقرار شده است . کد زیر را برای event مربوط به کلیک دکمه Send بنویسید :

```
Tcpserver.SendData txtSendData.text
```

- اگر داده ای از طرف Client بیايد رویداد DataArrival اجرا می شود . کد زیر را برای این رویداد بنویسید :

```
(As Long Private Sub tcpserver_DataArrival(ByVal bytesTotal
Dim strData As String
strData tcpserver.GetData
```

```
txtReceiveData.Text = strData  
End Sub
```

- کد زیر را برای رویداد Form_Unload بنویسید :

```
Tcpserver.Close
```

مراحل ساخت یک TCP Client بصورت زیر است :

- یک کنترل WinSock در فرم قرار دهید و نام آنرا tcpclient بگذارید .
- دو textbox با نامهای txtsend و txtreceive و نیز یک دکمه با نام send در فرم قرار دهید .
- یک دکمه با نام connect در فرم قرار دهید .
- کد زیر را برای متد Form_Load بنویسید :

```
tcpclient.RemoteHost="yourservername"x  
tcpclient.RemotePort=1000
```

- کد زیر را برای رویداد کلیک شدن دکمه connect بنویسید :

```
tcpclient.Connect
```

- کد زیر را برای رویداد کلیک شدن دکمه send بنویسید :

```
tctclient.SendData txtsend.Text
```

- کد زیر را برای رویداد DataArrival بنویسید :

```
(As Long Private Sub tcpclient_DataArrival(ByVal bytesTotal  
Dim strData As String  
strData tcpclient.GetData  
txtreceive.Text = strData  
End Sub
```

- کد زیر را برای رویداد Form_Unload بنویسید :

```
Tcpclient.Close
```

کدهای فوق یک سیستم Client-Server ساده را نشان می دهد . فایل exe هر دو برنامه را بسازید و آنها را اجرا کنید تا بتوانید سیستم خود را تست کنید .

5 – پذیرفتن بیش از یک تقاضای اتصال : Server ای که در بالا ساخته شد تنها می تواند تقاضای یک اتصال را بپذیرد . با استفاده از ایجاد یک آرایه از کنترل WinSock می توان چندین تقاضای اتصال را پذیرفت . برای اینکار کافی است یک کپی (instance) از کنترل بسازیم (با تنظیم خاصیت Index) و متد Accept را برای instance جدید بکار ببریم . فرض کنید یک کنترل WinSock با نام sckServer در فرم داریم که خاصیت Index آنرا صفر قرار داده ایم . همچنین یک متغیر intMax از نوع Long تعریف می کنیم که تعداد اتصالات همزمان به Server را نگه می دارد . در event مربوط به Form_Load کد زیر را بنویسید :

```
intMax=0
sckServer(0).LocalPort=1000
sckServer(0).Listen
```

هر بار که تقاضاي یک اتصال مي رسد کد ابتدا تست مي کند که مقدار Index چقدر است . اگر مقدار Index صفر باشد متغير intMax يکي افزايش مي يابد و از intMax براي ساخت یک instance جديد از کنترل استفاده مي شود . حال از اين instance براي پذيرفتن تقاضاي اتصال استفاده مي گردد . براي اينکار کد زیر را براي رویداد ConnectionRequest بنویسید :

```
(Integer, ByVal requestID As Long Private Sub sckServer_ConnectionRequest(Index As
If Index = 0 Then
1 + intmax = intmax
Load sckServer(intmax)x
0 = sckServer(intmax).LocalPort
sckServer(Index).Accept requestID
End If
End Sub
```

6 – ایجاد اتصال UDP : ساخت یک برنامه UDP ساده تر از برنامه هاي TCP است زیرا پروتکل UDP به اتصال نیاز ندارد . در برنامه TCP بالا یک کنترل WinSock بایستی حتماً Listen مي کرد و یک کنترل دیگر یک اتصال را توسط متد Connect ایجاد نمود . در عوض پروتکل UDP نیازی به اتصال ندارد . براي ارسال داده بین دو کنترل WinSock سه مرحله بایستی انجام شود :

- پارامتر RemoteHost برابر نام کامپیوتر مقابل است .
 - پارامتر RemotePort برابر پارامتر LocalPort کامپیوتر مقابل
 - استفاده از متد Bind براي مشخص کردن LocalPort
- چون هر دو کامپیوتر از نظر ارتباط مساوي هستند ، این نوع برنامه ها را Peer-to-Peer گویند . براي نمونه از کد زیر براي ساخت یک برنامه chat استفاده مي کنیم :
- یک کنترل WinSock در فرم قرار دهید و نام آنرا udppeerA بگذارید .
 - خاصیت Protocol آنرا UDPProtocol قرار دهید .
 - دو textbox با نامهاي txtsend و txtreceive و نیز یک دکمه در فرم قرار دهید .
 - کد زیر را براي متد Form_Load بنویسید :

```
udppeerA.RemoteHost="nameofpeerB"x
udppeerA.RemotePort=1001
1002 udppeerA.Bind
```

- کد زیر را براي event مربوط به کلیک دکمه بنویسید :

```
udppeerA.SendData txtsend.text
```

- کد زیر را براي رویداد DataArrival بنویسید :

```
Dim strData as String
strData udppeerA.GetData
txtreceive.Text=strData
```

برای ساخت UDP peerB مشابه مراحل بالا عمل کنید فقط خاصیت RemoteHost آنرا نام کامپیوتر PeerA و خاصیت RemotePort آنرا 1002 و خاصیت Bind آنرا 1001 قرار دهید .

کنترل WinSocket - قسمت دوم

بررسی خواص کنترل WinSocket :

ByteReceived : مقدار داده دریافت شده (موجود در بافر receive) را نشان می دهد . توسط متد GetData می توان این داده را دریافت نمود .

LocalHostName : نام ماشین محلی را نشان می دهد . این پارامتر فقط خواندنی است .

LocalIP : آدرس IP ماشین محلی را بصورت یک string برمی گرداند . این پارامتر فقط خواندنی است .

LocalPort : برای خواندن و یا تنظیم شماره پورت محلی بکار می رود .

Protocol : برای خواندن و یا تنظیم پروتوکل مورد استفاده توسط کنترل WinSocket بکار می رود .

RemoteHost : برای خواندن و یا تنظیم نام یا آدرس IP ماشین راه دور بکار می رود .

RemoteHostIP : آدرس IP ماشین راه دور را برمی گرداند :

۱- برای برنامه های Client بعد از زمانیکه یک اتصال توسط متد Connect پذیرفته شد ، این خاصیت حاوی آدرس IP ماشین راه دور است .

۲- برای برنامه Server ، بعد از آمدن یک Connection Request این خاصیت شامل آدرس IP ماشین راه دور است .

۳- در زمان استفاده از پروتکل UDP بعد از اینکه رویداد Arrival Data رخ داد این خاصیت حاوی آدرس IP ماشینی است که داده را فرستاده .

RemotePort : برای خواندن و یا تنظیم شماره پورت ماشین راه دوری که می خواهید به آن متصل شوید بکار می رود .

SocketHandle : مقداری را برمی گرداند که مرتبط با سوکتی است که کنترل WinSocket را مدیریت می کند و برای ارتباط با لایه WinSocket بکار می رود . این پارامتر فقط خواندنی است و تنها برای ارسال به API های WinSocket طراحی شده است .

State : وضعیت کنترل WinSocket را نشان می دهد . وضعیتهای ممکن برای State عبارتند از :

۱- sckClosed : اتصال بسته است .

۲- sckOpen : اتصال باز است .

۳- sckListening : حالت گوش دادن به پورت

۴- sckConnectionPending : معلق شدن اتصال

۵- sckResolvingHost : تصمیم گیری در مورد میزبان

۶- sckHostResolved : در مورد میزبان تصمیم گیری شد .

۷- sckConnecting : حالت برقراری ارتباط

۸- sckConnected : ارتباط برقرار شد .

۹- sckClosing : حالت قطع اتصال

۱۰- sckError : حالت خطا

بررسی متدهای کنترل WinSocket :

متد Accept : تنها برای برنامه های TCP Server بکار می رود . این متد برای پذیرفتن یک اتصال در زمان مدیریت رویداد ConnectionRequest استفاده می شود .

متد Bind : این پارامتر LocalPort و LocalIP یک اتصال را مشخص می کند .

متد Close : برای بستن یک اتصال TCP و یا بستن یک listening socket بکار می رود .

متد GetData : بلوک جاری داده دریافت شده را گرفته و آنرا در متغیری از نوع Variant ذخیره می کند . شکل کلی این متد بصورت زیر است :

```
WinSocket.GetData data[,type][,maxlen]x
```

که data داده دریافتی است . اگر داده کافی موجود نباشد data برابر empty خواهد بود .

type نوع داده دریافتی است که می تواند مقادیر زیر باشد :

vbString - vbInteger - vbLong - vbSingle - vbDouble - vbDate - vbBoolean - vbError - vbByte
vbArray+vbByte

maxlen حداکثر سایز را در زمان دریافت یک byte Array و یا یک string مشخص می کند .

متد GetData در رویداد Data Arrival استفاده می شود که این رویداد یک پارامتر با نام TotalBytes دارد . اگر maxlen ای که شما تعیین کرده اید کمتر از TotalBytes باشد پیغام هشدار شماره ۱۰۰۴۰ دریافت می کنید بدین معنی که بایتهای باقیمانده گم خواهند شد .

متد Listen : یک سوکت می سازد و آنرا در حالت Listen قرار می دهد . این متد تنها در اتصالات TCP بکار میرود .
متد PeekData : مشابه GetData است با این تفاوت که داده را از صف ورودی حذف نمی کند . این متد تنها برای اتصالات TCP بکار می رود .

متد SendData : برای ارسال داده به کامپیوتر راه دور بکار می رود .

بررسی event های کنترل WinSock :

رویداد Close : زمانی رخ می دهد که کامپیوتر راه دور اتصال را ببندد .

رویداد Connect : بعد از اینکه یک اتصال به Server ایجاد شد روی می دهد . شکل کلی آن بصورت زیر است :

```
Boolean)x Private Sub WinSock_Connect(Occurred As
```

که پارامتر Occurred دو مقدار دارد : اگر True باشد یعنی اتصال Fail شده است و اگر False باشد یعنی اتصال با موفقیت انجام شده است .

با رویداد Connect می توانید error هایی که در زمان فرایند باز کردن اتصال برگردانده شده را چک کنید .

رویداد ConnectionRequest : زمانی رخ می دهد که یک کامپیوتر راه دور تقاضای یک اتصال را بدهد . این رویداد فقط برای برنامه های TCP Server بکار می رود .

رویداد DataArrival : زمانی رخ می دهد که داده جدیدی بیاید .

رویداد Error : زمانی رخ می دهد که یک خطا در فرایند ارتباط رخ دهد (مثلاً Failed to Connect و یا Failed to Send) .
شکل کلی آن بصورت زیر است :

```
as String,scode as Long,source as String,helpfile as Private WinSock_Error(number as Integer,description  
Long,canceldisplay as Boolean)x String,helpcontext as
```

number شماره کد خطا است .

description توضیحی در مورد خطا است .

source توصیف منبع خطا

canceldisplay : مشخص می کند آیا پیغام خطای پیش فرض نشان داده شود یا نه

رویداد SendComplete : زمانی رخ می دهد که یک عمل Send تکمیل شده باشد .

رویداد SendProgress : زمانی رخ می دهد که کنترل شروع به ارسال داده نماید . شکل کلی آن بصورت زیر است :

```
bytesRemaining As Long)x ,WinSock_SendProgress (bytesSent As Long
```

که bytesSent تعداد بایتهای ارسال شده و bytesRemaining تعداد بایتهای باقیمانده است .

TAPI در ویژوال بیسیک - مقدمه

TAPI چیست ؟

TAPI یا Telephony API یک کتابخانه استاندارد برای کار با مودم و نوشتن برنامه های تلفنی می باشد . برای نمونه می توان از برنامه های Phone Dialer (شماره گیر تلفن) ، برنامه شبکه سازی تلفنی (Dialup Networking) ، برنامه تشخیص پالس مودم برای ضبط اطلاعات وارد شده از طرف کاربران و کاربردهای دیگر در این زمینه نام برد . این کتابخانه به شما کمک می کند تا بدون درگیر شدن با برنامه نویسی سخت افزار مودم و درایور آن بطور مستقیم بتوانید برنامه های کاربردی در این زمینه بنویسید .

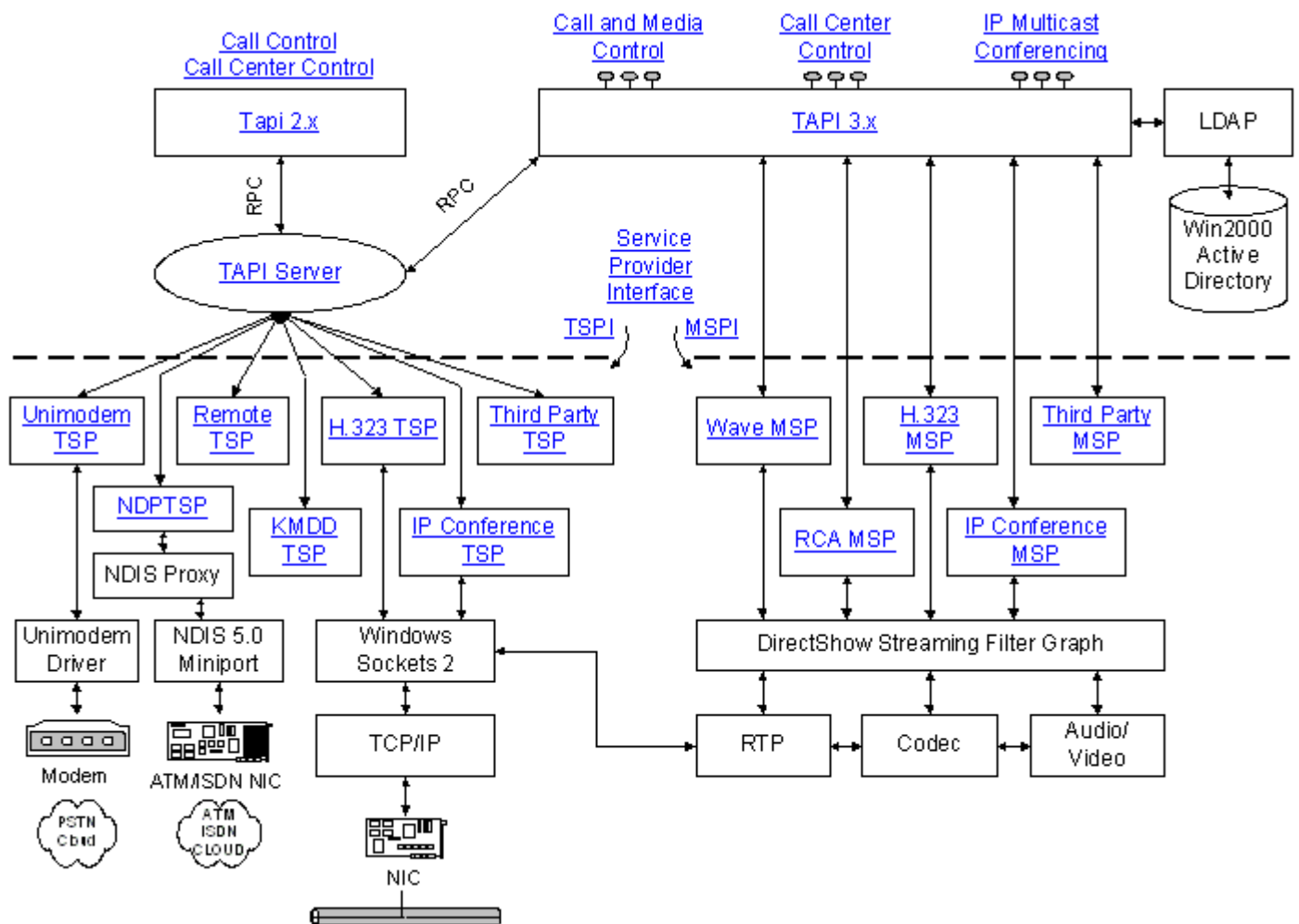
TAPI در ویژوال بیسیک - بخش اول

مروری بر Microsoft Telephony :

Telephony امکان مجتمع سازی کامپیوترها با دستگاههای ارتباطی و شبکه ها را فراهم نموده است . معمولاً دستگاه ارتباطی یک مودم و خط ارتباطی نیز شبکه PSTN (شبکه عمومی تلفن سوئیچینگ) می باشد . برخی از کاربردهای Telephony عبارتند از :

- ۱ - کنفرانسهای مالتی مدیا بصورت Multicast
- ۲ - VoIP
- ۳ - مرکز پاسخ گویی اتوماتیک
- ۴ - تماس تلفنی از طریق کامپیوتر روی شبکه PSTN

دیagram زیر معماری Microsoft Telephony را نشان می دهد :



برنامه های TAPI :

برای نوشتن برنامه های کاربردی با استفاده از TAPI بایستی ابتدا در مورد سطح سرویسی که می خواهیم ارائه دهیم تصمیم گیری کنیم . برای مثال برای نوشتن یک برنامه شماره گیر تلفن نیاز به استفاده کامل از TAPI نیست و می توان از قابلیت های خود ویندوز در این زمینه استفاده کرد (Assisted Telephony) . در بخش های بعدی در مورد سطوح مختلف سرویس در TAPI بیشتر صحبت خواهیم کرد .

دومین مطلبی که باید مورد توجه قرار داد اینست که می خواهیم از TAPI 2.x استفاده کنیم یا از TAPI 3.x . تفاوت این دو آنست که TAPI ورژن ۲ یک API بر مبنای C است در حالیکه ورژن ۳ آن بر مبنای تکنولوژی COM می باشد . در بخش های بعدی مطالب بیشتری در مورد تفاوت های این دو نسخه بیان خواهیم کرد .
بخش های اصلی یک برنامه کامل TAPI عبارتند از :

۱ - TAPI Initialization : شامل load کردن dll TAPI ، اتصال به Server TAPI ، مذاکره در مورد ورژن TAPI و برپاسازی سیستم اطلاع رسانی event می باشد .

۲ - Session Control : مقداردهی اولیه ، دریافت و کنترل تماسها

۳ - Device Control : دریافت و تنظیم اطلاعات دستگاه

۴ - Media Control : تشخیص و یا تولید تونها و ارقام ، کنترل stream

دیاگرام زیر این مراحل را بهتر نشان می دهد :

TAPI Initialization

Load TAPI DLL.
Connect to TAPISVR.
TAPI version negotiation.
Event notification setup.

Session Control

Initiate, receive, and manipulate calls.



Device Control

Get and set device information.

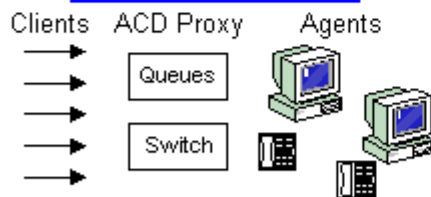


Media Control

Detect or generate tones and digits.
Stream manipulation.

Advanced Communications

Call Center Control



IP Multicast Multimedia Conferencing



TAPI Shutdown

Deallocate resources.

در بخشهای بعدی در مورد هر یک از این مراحل بیشتر صحبت خواهیم کرد .

TAPI در ویژوال بیسیک - بخش دوم

مقداردهی اولیه TAPI :

عملکرد درست اجزای TAPI نیاز به برپاسازی محیط ارتباطی روی کامپیوتر مورد نظر دارد . مراحل این امر عبارتند از :

۱ - نصب TAPI : زمانیکه سخت افزار و یا نرم افزار برای اولین بار به کامپیوتر اضافه می شود انجام می گیرد . جزئیات کار به سیستم عامل و نرم افزار بستگی دارد .

۲ - مقداردهی ابتدائی : ساخت اشیا و مسیرهای ارتباطی

۳ - مذاکره در مورد ورژن TAPI : برای اطمینان از اینکه اجزای TAPI قادر به تبادل داده ها باشند .

۴ - استخراج اطلاعات منابع : بدست آوردن اطلاعاتی در مورد دستگاهی که می توان از آن در برنامه TAPI مورد نظرم استفاده نمود .

۵ - Event notification : برپاسازی سیستم اطلاع رسانی event

TAPI در ویژوال بیسیک - بخش سوم

مقداردهی اولیه TAPI در ویژوال بیسیک :

از منوی Project گزینه References را انتخاب کرده و از لیست مربوطه مورد Microsoft TAPI 3.0 Type Library را انتخاب کنید .

حال وارد بخش کد نویسی فرمتان شوید و متغیر objTAPI را بصورت زیر تعریف کنید :

```
Dim objTapi As TAPI
```

سپس در بخش مربوط به Load Form شی objTAPI را بصورت زیر ایجاد می کنیم :

```
Set objTapi = New TAPI
```

همانطور که در بخشهای قبلی گفته شد ، قبل از فراخوانی هر تابع TAPI ابتدا بایستی آنرا مقداردهی اولیه کنیم . برای مقداردهی اولیه کردن شی TAPI عبارت زیر را بنویسید :

```
Call objTapi.Initialize
```

TAPI در ویژوال بیسیک - بخش چهارم

انتخاب یک آدرس :

کد زیر نشان می دهد که چگونه می توان با استفاده از شی TAPI در ویژوال بیسیک منابع تلفنی در دسترس را برای یک آدرس که بتواند یک مجموعه مشخص از نیازها را مدیریت کند ، بررسی کرد .
توجه داشته باشید که قبل از انجام این کار بایستی عمل مقداردهی اولیه TAPI را که در بخش قبل بررسی شد ، انجام دهید .

نکته : در کد زیر عمل error checking انجام نگرفته است و برای استفاده از کد زیر در برنامه های واقعی بایستی بخش بررسی خطا را به آن اضافه کنید .
۱ - تعریف یک شی آدرس و یک شی مجموعه آدرس :

```
Dim gobjAddress As ITAddress  
As ITCollection Dim objCollAddresses
```

۲ - تنظیم شی objCollAddress بعنوان یک مجموعه آدرس از شی objTapi :

```
Set objCollAddresses = objTapi.Addresses
```

۳ - پیدا کردن آدرسی که بتواند از واسط مورد نظر ما پشتیبانی کند :

```
bFound = False  
objCollAddresses.Count For indexAddr = 1 To  
objCollAddresses.Item(indexAddr)x = Set objCrtAddress  
Set objMediaSupport = objCrtAddress  
objAddressCapabilities = objCrtAddress Set  
  
nSelectedType ) x )If objMediaSupport.QueryMediaType  
bFound = True  
End If  
  
Nothing = Set objAddressCapabilities  
Set objMediaSupport = Nothing  
Nothing = Set objCrtAddress  
  
If bFound = True Then Exit For  
Next indexAddr
```

در صورتیکه آدرس مورد نظر پیدا شود برنامه از حلقه خارج شده و gobjAddress یک آدرس قابل استفاده خواهد بود :

```
objcollAddresses.Item(indexAddr)x = Set gobjAddress
```

TAPI در ویژوال بیسیک - بخش پنجم

انجام Event Handling در TAPI :

کد زیر شامل یک event handler ساده برای TAPI ، رجیستر کردن واسط event ، تنظیم فیلتر event و رجیستر کردن تمام فراخوانیهای دادن اخطار است . هدف اصلی از این کد اینست که مطمئن شویم بخشی از TAPI که event ها را دریافت می کند پردازشی را قبل از انتقال به بخشهای دیگر انجام دهد .

تعاریفها :

```
TAPI Dim WithEvents gobjTapiWithEvents As  
Attribute gobjTapiWithEvents.VB_VarHelpID = -1  
glRegistrationToken As Long Dim
```

```
Const TAPI3_CALL_EVENTS = TE_CALLMEDIA Or  
TE_CALLNOTIFICATION Or TE_CALLSTATE
```

تنظیم eventfilter بصورتیکه تمام event های تعریف شده برای TAPI را بپذیرد :

```
TAPI3_CALL_EVENTS = objTapi.EventFilter
```

رجیستر کردن event ها :

```
Set gobjTapiWithEvents = objTapi  
Boolean, fMonitor As Boolean Dim fOwner As  
Long Dim IMediaTypes As Long, ICallbackInstance As
```

```
fOwner = True  
fOwner = True  
fMonitor = False  
TAPIMEDIATYPE_AUDIO = IMediaTypes  
ICallbackInstance = 1
```

```
,gobjTapi.RegisterCallNotifications(gobjAddress,fMonitor = glRegistrationToken  
fOwner,IMediaTypes,ICallbackInstance)x
```

TAPI در ویژوال بیسیک - بخش ششم

انتخاب یک ترمینال :

+ قبل از اینکه یک ترمینال را برای برقراری ارتباط انتخاب کنید بایستی TAPI Initialization و عمل انتخاب آدرس را انجام داده باشید .

ابتدا یک متغیر از نوع ITBasicCallControl (واسط کنترل تماس) تعریف می کنیم :

```
Dim objCallControl As ITBasicCallControl  
objCallControl = gobjReceivedCallInfo Set
```

سپس یک متغیر از نوع ITTerminalSupport (کوئری از شی آدرس) تعریف می کنیم :

```
Dim objTerminalSupport As ITTerminalSupport  
objTerminalSupport = gobjAddress Set
```

سپس متغیر ترمینال را تعریف کرده و توسط شی objTerminalSupport یک ترمینال را برای آن استخراج می کنیم :

```
Dim objTerminal As ITTerminal  
objTerminalSupport.GetDefaultStaticTerminal(IMediaType, dir)x = Set objTerminal
```

در اینجا دیگر نیازی به شی objTerminalSupport نیست بنابراین آنرا آزاد می کنیم :

```
Set objTerminalSupport = Nothing
```

سپس نیاز به تعریف شی objStreamControl برای کنترل ترمینال است :

```
Dim objStreamControl As ITStreamControl  
objStreamControl = objCallControl Set
```

در صورتیکه این شی ایجاد شود ، به ازای استریم های موجود در ITCollection امکان ایجاد ترمینال در یک حلقه for بررسی می شود و ترمینال مناسب انتخاب می گردد :

```
If Not (objStreamControl Is Nothing) Then  
objITCollStreams As ITCollection Dim
```

```
objStreamControl.Streams = Set objITCollStreams
```

```
ITStream Dim nIndex As Long, objCrtStream As
```

```
For nIndex = 1 To objITCollStreams.Count  
objITCollStreams.Item(nIndex)x = Set objCrtStream  
Then If objCrtStream.MediaType = IMediaType  
If objCrtStream.Direction = dir Then  
objCrtStream.SelectTerminal(objTerminal)x Call  
End If  
End If  
objCrtStream = Nothing Set  
Next nIndex
```

```
Nothing = Set objITCollStreams  
Set objStreamControl = Nothing  
End If
```

ایجاد یک تماس (Make a Call) :

+ قبل از این بخش بایستی مراحل TAPI Initialization و عمل انتخاب آدرس انجام شده باشد .
این بخش برای ایجاد یک شی تماس ، بررسی و مشخص کردن استریمی که با این تماس در ارتباط است ، انتخاب و
ایجاد ترمینالهای مناسب و کامل کردن ارتباط استفاده می شود .
قبل TAPI Initialization و عمل انتخاب آدرس و انتخاب ترمینال انجام شده باشد .
در ابتدا با استفاده از متد CreateCall یک شی تماس ساخته می شود :

```
nSelectedType,IMediaTypes)x ,gobjOrigAddress.CreateCall(strDestAddress = Set gobjCall
```

سپس در اینجا بایستی کدی که در بخش اول این درس برای انتخاب ترمینال نوشته شد آورده شود :

```
{  
Select Terminal Code  
}
```

سپس بایستی دستور Connect اجرا شود :

```
gobjCall.Connect (False)x
```

False بدین معناست که ارتباط بصورت آسنکرون برقرار می شود .

TAPI در ویژوال بیسیک - بخش پایانی

دریافت یک تماس :

کد زیر برای یافتن و یا ایجاد یک ترمینال مناسب برای دریافت یک تماس بکار می رود . بایستی توجه داشته باشید که قبل از اجرای کد زیر بایستی مراحل مقداردهی اولیه ، انتخاب یک آدرس و رجیستر کردن event ها را انجام دهید . همچنین در کد زیر بایستی مرحله انتخاب ترمینال را نیز انجام دهید . توجه داشته باشید که در کد زیر متغیر pEvent یک اشاره گر برای واسطه ITCallNotificationEvent است که توسط TAPI به event Handler داده می شود :

```
If TapiEvent = TE_CALLNOTIFICATION Then
objCallNotificationEvent As ITCallNotificationEvent Dim
objCallNotificationEvent = pEvent Set
ITCallInfo Dim gobjReceivedCallInfo As
Set gobjReceivedCallInfo = objCallNotificationEvent.Call
objCallControl As ITBasicCallControl Dim
gobjReceivedCallInfo = Set objCallControl
objCallControl.Answer
End If
```