به نام خدا

استفاده از Thread ها در دلفی

اساس کار ویندوز در اجرای برنامه ها بر روی Thread ها است ، هنگامی که برنامه ای اجرا می شود ، کلیه کـدها و فایلهای منبع (Resource) آن در یک یا چند Thread قرار می گیرند ، درواقع هرProcess در ویندوز دارای یک یا چند Thread است که اطلاعات Process درآنها قرار دارند و بر روی حافظه موقت بارگذاری شده اند ... ، برنامه ها میتوانند در زمان اجرای خود Thread هایی ساخته و سپس آنها را آزاد نمایند ، کار یک Process زمانی به پایـان می رسد که تمام ، Thread های آزاد شده باشند ، در غیر این صورت آن Process بسته نخواهد شد ، پـس بایـد در آزادسازی Thread ها دقت زیادی داشت ...

ساخت Thread و استفاده از آن در برنامه :

در دلفی کلاسی به نام TThread قرار دارد که امکان استفاده از Thread ها را فراهم می سازد ...

برای استفاده از یک Thread ابتدا باید یک نسخه از کلاس TThread را Create کرده و سپس از آن استفاده نمایید

برای ساخت یک Thread به صورت زیر عمل می کنیم :

```
Type
 MyThread = Class(TThread)
  private
    …
  protected
    ...
 end;
```

در کد بالا یک نوع از TThread را ساختیم ...

در یک Thread میتوانید Procedure ها و توابعی را قرار داده و از آنها استفاده نمایید ، توجه داشته باشید که بـرای فراخوانی توابع و متد ها باید از روش Synchronization (همزمان سازی) استفاده نمایید ، شاید بـا دیـدن ایـن کلمات احساس کنید که کار سختی در پیش دارید ، اما اصلا این طور نیست ، این کار با اجرای یک تابع سـاده امکـان پذیر است که در ادامه توضیح خواهیم داد ...

Thread ها یک Event اصلی به نام OnExecute دارند که مربوط به زمان اجرا شدن آنها است ، تمام کار Thread ها در همین Event انجام می شود ، کدی که در این رویداد می نویسید از زمـان اجـرا شـدن Thread تـا زمـان Terminate شدن آن اجرا خواهد شد.

نمی توان در یک Thread به صورت مستقیم با اشیای فرم ارتباط داشت ، چنین درخواستهایی از Thread باعث متوقف شدن کار آن خواهد شد و نتیجه مطلوبی نخواهید گرفت ... ، برای دسترسی به اشیاء باید از Procedure های جدا و Synchronize استفاده نمایید تا اختلالی در کار Thread به وجود نیاید ...

ابتدا باید توابع و متد ها را در یک Thread تعریف کرده و کـد مـورد نظـر را در آن بنویـسیم ، سـپس بـا تـابع Synchronize به روش زیر می توانیم آن را اجرا نماییم :

```
Synchronize( My Procedure );
```

طریقه ایجاد توابع و Procedure ها در Thread مانند سایر کلاسها (مثلا TForm) است ، برای مثال فرض کنید متدی داریم که در آن عمل Progress کردن یک ProgressBar را انجام می دهیم ، میتوانیم به صورت زیر این متد را تعریف نماییم :

```
Type
 MyThread = Class(TThread)
  private
   procedure doProgress;
  end;

Implementation

procedure doProgress;
begin
  Form1.ProgressBar1.Progress;
end;
```

ساخت و استفاده از Thread :

در بالا نحوه تعریف یک Thread را توضیح دادیم ، برای استفاده از آن باید یک متغیر با نام Thread تعریف شده تعریف نماییم و از آن استفاده کنیم ...

نکته دیگر این که متد Create برای Thread ها یک پارامتر به نام Suspended دارد که از نوع Boolean می باشد و به صورت پیشفرض دارای مقدار False است ...

این پارامتر مشخص میکند که آیا Thread در حالت متوقف ساخته شود یا اینکه بعد از ساخته شدن بلافاصله در حالت اجرا قرار گیرد (Thread بعد از ساخته شدن ، متد Execute اش اجرا خواهد شد ، با True کردن این پارامتر ، متد Execute را متوقف کرده و از اجرا شدن کدها جلوگیری می نماییم) ، بعد از اجرای Thread ممکن است نیاز باشد که برخی از خصوصیات آن را تغییر دهیم ، پس باید به این پارامتر مقدار True دهیم تا Thread ما بعد از ساخته شدن در حالت اجرا نباشد و امکان تغییر خاصیتهای آن وجود داشته باشد ...

برای مثال بهتر است که خاصیت FreeOnTerminate مربوط به Thread ساخته شده را True نماییم تا هنگام فراخوانی متد Terminate برای پایان کار Thread ، آن را آزاد کنیم ...

پس کد ما تا اینجا به شکل زیر درخواهد آمد :

```
var
 T : MyThread;
begin
 T := MyThread.Create(True);
 T.FreeOnTerminate := True;
 T.Resume;
end;
```

در کد بالا Thread ما ساخته شده و سپس خاصیت FreeOnTerminate آن True شده و سپس با استفاده از متد Resume به کار خود ادامه می دهد (از حالت Suspend بیرون می آید) ...

کد نویسی در رویداد OnExecute :

هر Thread ای که Create می کنید در حالت پیشفرض رویداد OnExecute را دارا می باشد ، برای این که آن را به دلخواه خود تغییر دهید باید آن را دوباره تعریف کنید ، برای تعریف این رویداد باید از قسمت Protection و واژه

Override استفاده نمایید تا رویداد قبلی Thread از بین رفته و رویداد جدیدی که ایجاد می کنید جایگزین شـــود ، پس کد شما به این صورت خواهد بود :

```
type
 MyThread = Class(TThread)
  private
  ...
  protected
   procedure Execute; override;
  end;
var

implementation

procedure MyThread.Execute;
begin

end;
```

اجرای Procedure های تعریف شده (Synchronization) :

برای فراخوانی Procedure هایی که تعریف کرده اید باید ار تابع Synchronize استفاده نمایید (در داخل رویداد OnExecute) ، برای مثال :

```
procedure MyThread.Execute;
begin
 Synchronize(doProgress);
end;
```

تابع Synchronize یک پارامتر دارد که از نوع TThreadMethod است و Procedure ای که در Thread تعریـــف کردید باید در آن قرار گیرد ...

مدیریت Thread :

برای استفاده از Thread با متدهای آن آشنایی داشته باشید ...

برای متوقف کردن یک Thread باید از متد Suspend استفاده نمایید ، و برای ادامه کار آن بایـــد از متـــد Resume استفاده کنید :

```
MyThread.Suspend;
...
MyThread.Resume;
```

برای این که بفهمید آیا Thread ما در حالت Suspend است یا خیر میتوانید از خاصیت Suspended استفاده نمایید ، این خاصیت یک مقدار Boolean دارد که مشخص می کند Thread مورد نظر Suspend شده یا خیر :

```
var
 isSuspended : Boolean;
begin
 isSuspended := MyThread.Suspended;
end;
```

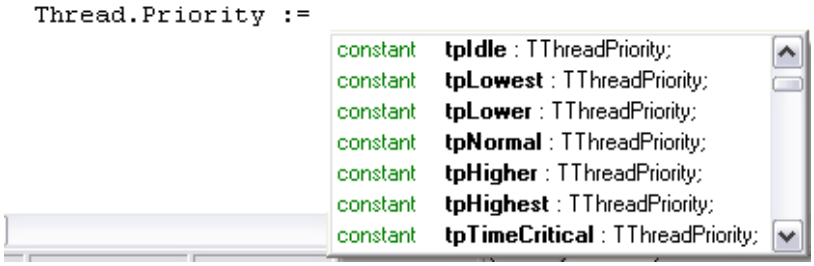
اگر می خواهید که کار یک Thread را کاملا متوقف نمایید ، می توانید از متد Stop استفاده کنید ، در ایـــن صـــورت برای اجرای دوباره Thread باید از متد Execute استفاده نمایید :

```
MyThread.Stop;
...
MyThread.Execute;
```

خاصیت Priority :

اگر در محیط ویندوز برنامه Task Manager را اجرا نموده و به قسمت Process بروید لیست Process های در حال اجرا را می بینید ، اگر بر روی هر کدام از این Process ها راست کلیک نمایید گزینه ای به نام Set Priority می بینید که مقادیری مثل High ، Normal یا Low و ... دارد ، Priority مشخص می کند که در زمان الویت بنـــدی اجـــرای Thread ها در CPU ، کدام یک ارجعیت دارند و به CPU ابتدا باید به درخواست کدام یک از آنهـــا جـــواب دهـــد ، هرچه مقدار Priority یک Thread مقدار بالاتری داشته باشد ، الویت بیشتری خواهد داشت و عملیاتش زودتر انجام خواهد شد ...

می توانید برای Threadخود این خاصیت را تنظیم نمایید ، این خاصیت از نوع TThreadPriority مـــی باشـــد ، در تصویر زیر مقادیری که می توانید به عنوان Priority قرار دهید مشخص شده است :

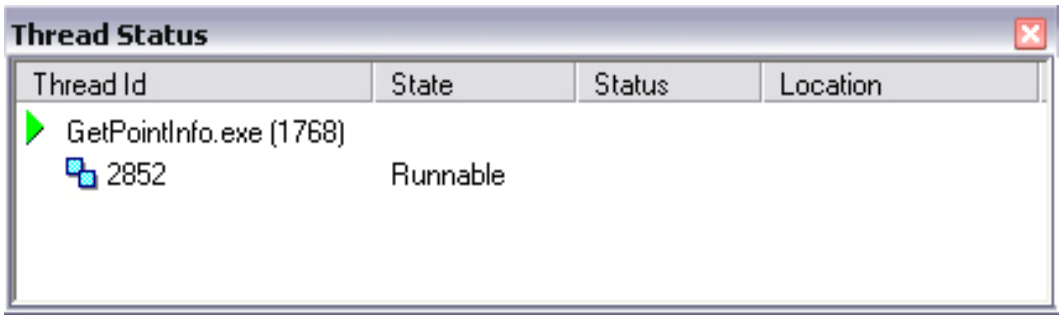


ThreadID :

این خاصیت شناسه ای برای Thread شما است که هم در زمان ساخت (در زمان Debug) و هـــم در زمـــان اجـــرای برنامه می توانید از آن استفاده نمایید ، هنگامی که قصد Debug کردن برنامه خود در محیط دلفی را دارید ، پـــس از اجرا کردن برنامه اگر از منوی View گزینه Debug Windows و سپس گزینه Threads را انتخاب نمایید ، پنجـــره ای باز شده و لیستی از Thread های در حال اجرا در برنامه شما را نمایش میدهد که هرکدام از آنها دارای شناسه ای به نام ThreadID هستند ، با داشتن این مقدار می توانید Thread مورد نظر خود را در این پنجره پیدا نمایید ...

این مقدار را می توانید به روش زیر بدست آورید :

```
var
 MyThreadID : Cardinal;
begin
 MyThreadID := MyThread.ThreadID;
end;
```



خروج از Thread و آزاد کردن آن :

رویداد OnTerminate :

Thread یک رویداد دیگر دارد که میتوانید آن را مانند OnExecute تنظیم نمایید و تغییر دهید ، این رویداد زمانی اتفاق می افتد که Thread مورد نظر Terminate شود ، اما متد دیگری به نام DoTerminate وجود دارد که این رویداد را اجرا میکند بدون اینکه Thread مورد نظر Terminate شود (یا شده باشد) ، با متد ...

اگر یادتان باشد ، ما در هنگام ساخت Thread مقدار خاصیت FreeOnTerminateآن را True کردیم پس اگر آن را Terminate نماییم ، آزاد (Free) خواهد شد ، با متد Terminate میتوانید به کار یک Thread پایان دهید :

```
MyThread.Terminate;
```

در برخی موارد ممکن است Thread مورد نظر Terminate نشود ! ، درواقع هنگامی که ما متد Terminate را اجرا می کنیم ، کار عمده ای انجام نمی شود ، بلکه فقط خاصیت Terminated مربوط به Thread بـــر روی True تنظـــیم می شود ! ، درواقع این عمل به Thread اعلام میکند که به زودی ممکن است کارش پایان یابد ، باید برای پایان کـــار Thread ، بعد از اجرا این متد با استفاده از متد Exit (مربـــوط به Thread نیست) از رویداد OnExecute خـــارج شد (دستور Exit در دلفی برای خروج از یک رویه (تابع یا ...) به کار می رود) :

```
procedure MyThread.Execute;
begin
 Synchronize(doProgress);
 …
 MyThread.Terminate;
 Exit;
end;
```

توجه داشته باشید که در قسمتهای مختلف کد باید چک کنید که آیا Terminated مربوط بـــه Thread مقـــدار True دارد یا خیر و اگر مقدار True داشت با استفاده از دستور Exit از رویداد خارج شوید :

```
procedure MyThread.Execute;
begin
 if MyThread.Terminated then
   Exit;
 …
 if MyThread.Terminated then
   Exit;
 …
 MyThread.Terminate;
 Exit;
end;
```

علاوه بر موارد بالا می توانید از متد WaiteFor استفاده نمایید ، متد WaiteFor تا زمانی که کار Thread به پایـــان برسد کارش تمام نخواهد شد ، درواقع می توانید کد Exit را بعد از این متد بنویسید تا خیالتان از بابت پایـــان کـــار Thread راحت شود ! :

```
procedure MyThread.Execute;
begin
 if MyThread.Terminated then
   Exit;
 …
 if MyThread.Terminated then
   Exit;
 …
 MyThread.Terminate;
 MyThread.WaiteFor;
 Exit;
end;
```

این مقاله هم به پایان رسید ، امیدوارم مفید بوده باشه ...

http://www.SayehGroup.com

محمود مهری

تابستان ۱۳۸۶