الريجستري

في

دلفي

تأليف:

أحمد الميّاحي

الإهداء...

إلى بلدي العزيز الغالي...

أهدي إليك كتابي هذا

لعلّك تقبله منّي كهدية متواضعة

إلى العراق

أهدي كتابي هذا

الفهرس

مقدمة

لا شكّ أنَّ الريجستري (Registry) يلعب دوراً أساسياً في نظام التشغيل ويندوز! ماذا تتصور لو لم يكن الريجستري؟ بالتأكيد . . . أنَّ عمل تخزين واستدعاء البيانات سيواجه صعوبة حقيقية ومشكلة عالقة.

لا يخفى عليكم أن Delphi تتيح لكم إمكانية التعامل مع الريجستري بصورة سهلة وقوية في نفس الوقت! فلا داعي لاستخدام دوال API الخاصة بالريجستري كما في C++ أو في Visual Basic! إنَّ دوال الريجستري الخاصة بدلفي اليوم تتيح لك إمكانية التعامل مع الريجستري بعيداً عن دوال API المعقّدة، أنت اليوم كمبرمج Delphi تفرض نفسك على كل مبرمجي العالم بأي لغةٍ كانوا يبرمجون! خصوصاً بعد إصدار Delphi 8.0 for the Microsoft .NET framework والتي دمجت لغتين في بيئة واحدة! بإمكانك البرمجة لـ .NET. اليوم باستخدام Delphi نفسها! فلا تتعب نفسك وتتعلم أحد لغات .NET. فـ Delphi هي .NET. وفي نفس الوقت هي VCL Application.

ستتعلم كيفية إنشاء البرامج التفاعلية لحفظ البيانات في الريجستري وستتعلم طريقة إنشاء برنامج (Background) وهو برنامج بسيط ورائع يوفر إمكانية إدراج صورة إلى شريط الأدوات في المستكشف وجهاز الكمبيوتر.

لقد حاولت جهد الإمكان أن يكون هذا الكتيّب دليلك الشامل إلى التعامل مع الريجستري باستخدام Delphi!! دروسٌ بسيطة جداً تعلّمك طريقة التعامل مع الريجستري.

وأخيراً . . .

إنَّ الإنسان غير معصوم فهو يخطئ ويصيب! فإن كان كتيبي هذا فيه نوع من الغموض أو فيه نوع من التقصير فلا تتردد بالاتصال بي على البريد الإلكتروني:

Mayahi_Ahmad@Yahoo.com
Mayahi_Iraq@hotmail.com
Mayahi2@paltalk.com
ICQ Number:

والسلام عليكم ورحمة الله وبركاته

ما هو الريجستري ؟

سؤال لطيف ووجيه في نفس الوقت! سأجيب على هذا السؤال بطريقة مفصّلة وواضحة جداً يفهمها القريب والبعيد!!

هناك برامج عديدة تستخدم الريجستري كأداة لتخزين البيانات التي تستخدمها بشكلٍ متكرر. مثلاً: تذكير اسم المستخدم، حفظ آخر مجلد تم الوصول إليه، وجميع ما هو من قبيل هذه الأشياء. وكل هذه الأشياء هي للتذكير.

لأمثّل بصورة أوضح! لنفترض أنَّ لديك البرنامج التالي:

كما هو موضّح فإنَّ المستخدم لابدّ أن يكتب "اسم المستخدم" و"كلمة المرور" لكي يستطيع الدخول إلى البرنامج. أما أنت كمبرمج فتريد حفظ "اسم المستخدم" في الـ Edit الخاص "باسم المستخدم" كلما فتح المستخدم البرنامج، يعني كبرامج الـ Messenger مثلاً msn Messenger والذي يحفظ "اسم المستخدم" في الـ Edit تلقائياً.

طبعاً هذا كمثال بسيط جداً جداً على تخزين المعلومات واستدعائها في الـ Edit وهناك عدّة برامج عملاقة تستخدم الريجستري كجزء أساسي من برنامجها. لكن نكتفي بهذا المثال البسيط ولا نعقّد الأمور أكثر من اللازم. والآن عرفنا أن الريجستري هو أداة لتخزين البيانات.

تاريخ الريجستري

إنَّ الريجستري حديث العهد نسبياً، أوّل مرة ظهر فيها الريجستري هي عندما ظهر نظام التشغيل Windows 95 وكان الهدف من وراء الريجستري هو توفير أداة فعّالة لحفظ البيانات والوصول إليها مجدداً. وقد وُجد الريجستري على أساس قاعدة بيانات تحتفظ بالبيانات المرسلة إليها من البرامج.

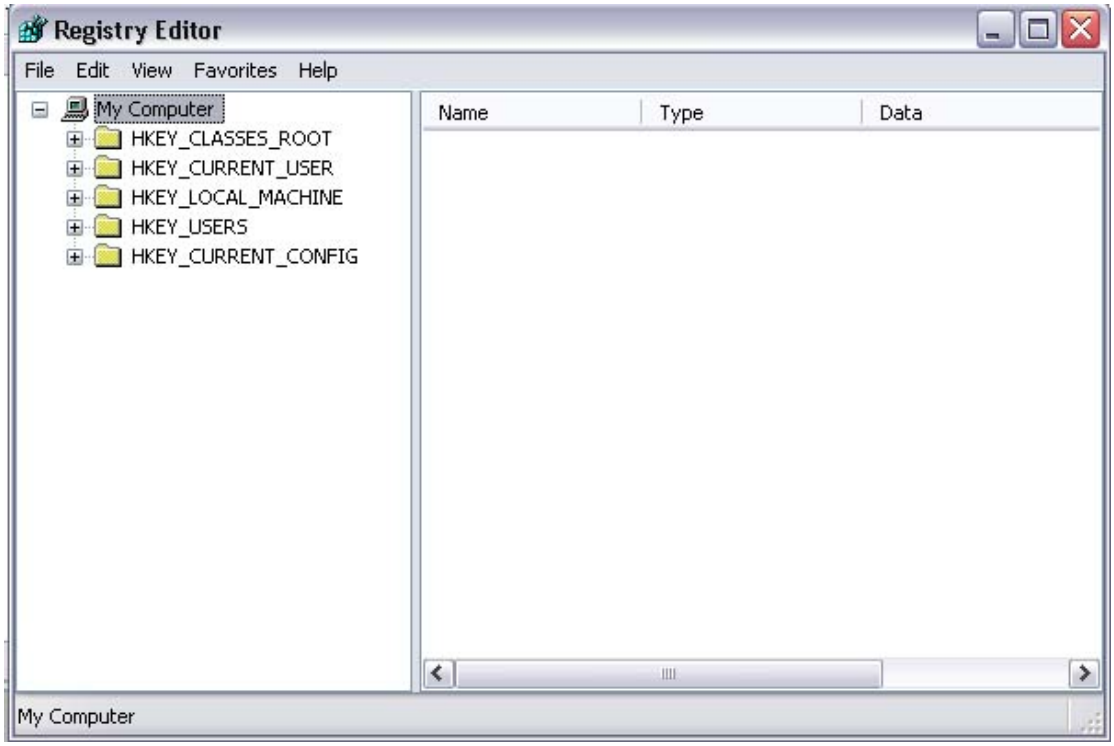
لكنك تستاءل الآن كيف يتم تخزين البيانات سابقاً؟

سابقاً كان يتم تخزين البيانات بطريقة ملف الـ ini كما في Windows 3.11 فهناك عدة ملفات خاصة بنظام التشغيل مثل System.ini وWin.ini حيث يتم حفظ البيانات في هذان الملفان، ولا يخفى عليك أنَّ ملفات الـ ini إلى الآن تستخدم ولكنها تستخدم لطرق خاصة مثلاً حفظ أسماء الملفات المفتوحة مؤخراً وغيرها من الأمور الأخرى التي لا تحتاج إلى ريجستري، وملفات الـ ini هذه هي ملفات خاصة مرتبطة بنفس البرنامج ولا علاقة لها بنظام التشغيل يعني بعبارة أخرى هي لا تستخدم ملفات الويندوز Win.ini أو System.ini.

من الدوافع التي أظهرت الريجستري هي عدم قدرة هذه الملفات على تخزين البيانات الكبيرة والتي تستخدم بكثرة! لأن ملف System.ini وWin.ini مختصين بنظام التشغيل ذاته وتقلع هذه الملفات عند إقلاع نظام التشغيل وبالتالي سوف تقلع جميع البيانات المكتوبة في هذه الملفات تلقائياً مما يسب بطئاً واضحاً في استنهاض نظام التشغيل نفسه وأيضاً في التعامل مع نظام التشغيل.

كيف أستخدم الريجستري ؟

إنَّ الريجستري (Registey) هو برنامج مرفق مع نظام التشغيل Windows وكما ذكرنا سابقاً فإنه أداة فعالة لتخزين البيانات واستدعائها من خلال البرامج. وإذا فتحنا الريجستري وذلك من خلال أبدأ > تشغيل > ثم كتابة الأمر RegEdit فسوف تظهر الشاشة التالية:



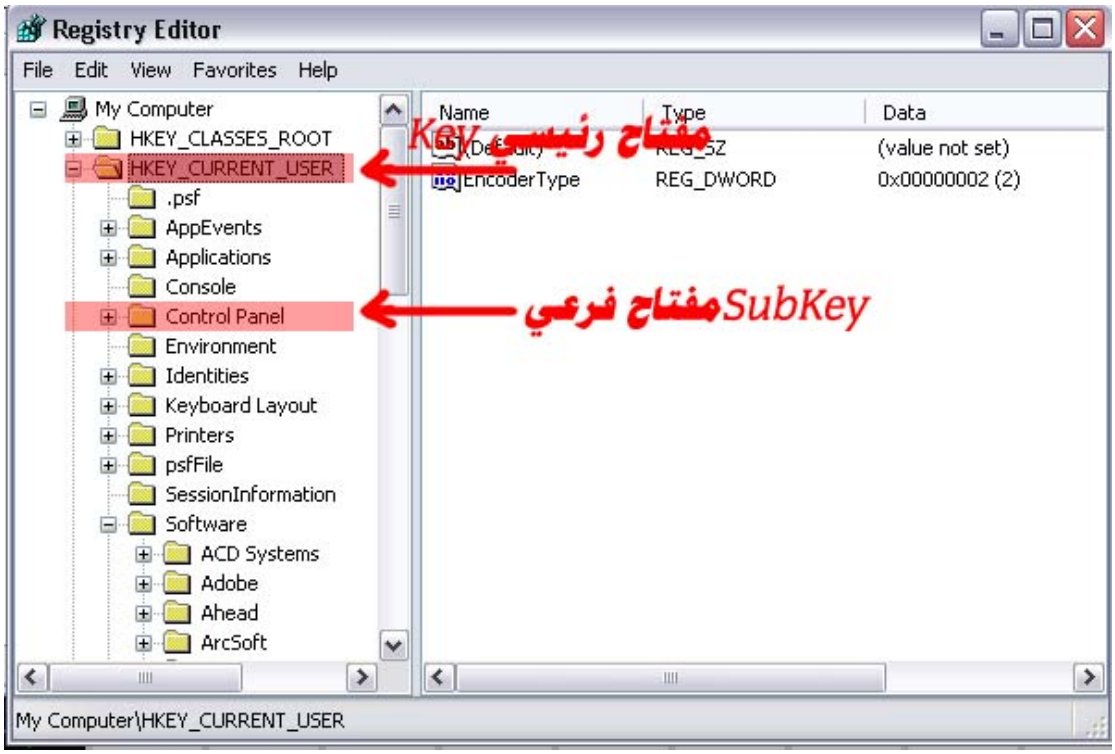
يتكون الريجستري من مفاتيح رئيسية ومفاتيح فرعية مصنّفة، وهذه المفاتيح متوفرة على الجهة اليسرى من الريجستري وكل هذه مفاتيح رئيسية مصنّفة حسب البيانات المخزّنة فيها مثلاً مفتاح HKEY_CLASSIC_ROOT وهو مفتاح رئيسي تتفرع منه مفاتيح فرعية خاصّة بلواحق الملفات مثلاً اللاحقة bmp والخاصة بملفات الـ Bitmap والتي تعمل تحت برنامج الرسام، فقد أدرج اسم هذه اللاحقة بالإضافة إلى مسار برنامج الرسام وعندما يصادف الـ Windows أي لاحقة بهذا الاسم يعني bmp فسوف ينتقل تلقائياً ويشغل برنامج الرسام ليظهر هذا الملف. . وإلخ.

أما بالنسبة للمفاتيح الرئيسية الأخرى فهي بهذا الشكل:

- HKEY_CLASSIC_ROOT: مفتاح رئيسي خاص بلواحق الملفات، جميع لواحق الملفات نظير bmp-doc-dfm وغيرها تحفظ في هذا المفتاح الرئيسي.
- HKEY_CURRENT_USER: مفتاح رئيسي يحتفظ بمعلومات البرامج الأخرى، مثلاً معلومات Windows ستجدها في هذا المفتاح الرئيسي، وهو مصنّف حسب المفاتيح، مثلاً Software يندرج تحته البرامج / الشركات المستخدمة في جهازك.

- HKEY_LOCAL_MACHINE: وهو مفتاح خاص بمعلومات جهازك وماذا يحتوي من قطعات وإلخ. . . من معلوماتٍ آخرى.
- HKEY_USERS: مفتاح رئيسي خاص بمستخدمي الكمبيوتر.
- HKEY_CURRENT_CONFIG: مفتاح خاص بهيئة الكمبيوتر.

الآن نأتي إلى المفاتيح الفرعية، أنظر إلى الشكل التالي:



ألقِ نظرة على المفاتيح الرئيسية وكيف تتفرع منها مفاتيح فرعية، المفاتيح الرئيسية تسمّى Key والمفاتيح الفرعية تسمى Sub Key.

وهناك قِيَم (Value) وهي موجودة في المفاتيح الرئيسية والمفاتيح الفرعية ويمكن للقيم أن تأخذ أيّاً من أنواع البيانات التالية:

- سلسلة حرفية (String).
- أرقام ثنائية (Binary).
- نوع أرقام (Dword).
- سلسلة متعددة (Multi-String Value).

❖ سلسلة توسّعية (expandable).

والقيم هي أساس الريجستري حيث جميع البيانات تخزّن في القيم، مثلاً نجعل قيمة بعنوان UserName تحفظ "اسم المستخدم" وهذا بالنسبة للقيم الأخرى، وللوضوح الأكثر أنظر إلى الشكل التالي:

وبناءً على ذلك فالقيمة هي الأساس والتي نستخدمها لتخزين البيانات ويمكن للقيمة أن تشمل نوع واحد في نفس الوقت فقط واسماً لا يتكرر، هذا كل ما في الريجستري، أما الأشياء البقية والاحترافية فلا يسع الوقت لذكرها هنا يمكنك مطالعة العديد من الكُتب سواءً العربية أو غيرها لكي تحصلوا على معلومات احترافية قيمة عن الريجستري، والآن نبدأ باسم اللّه لنخوض في التفاصيل وعن كيفية التعامل مع الريجستري وتخزين القيم فيه من خلال بيئة دلفي، طبعاً لا يفرق ماذا تستخدم من بيئة أنا شخصياً أستخدم Delphi 7 ولدي Delphi 8 أيضاً إلا أنني أستخدمه لبناء تطبيقات .NET فقط.

إخبار دلفي بأننا نريد استخدام الريجستري في برنامجنا

ليس من المنطقي كتابة معلومات البرنامج بواسطة ملف txt عادي وهذا ما يفعله الكثير من المبرمجين ولا أعلم لماذا لا يبحث هؤلاء المبرمجين عن طريقة مثلى غير الملفات النصية لتخزين بياناتهم؟ وأنا موقنٌ بأن الريجستري هو أفضل بكثير من الملفات النصية لتخزين البيانات! على كل حال فلست بصدد نقاش هؤلاء.

سأشرح هنا كيف تخبر Delphi بأنك تريد استخدام الريجستري في برنامجك.

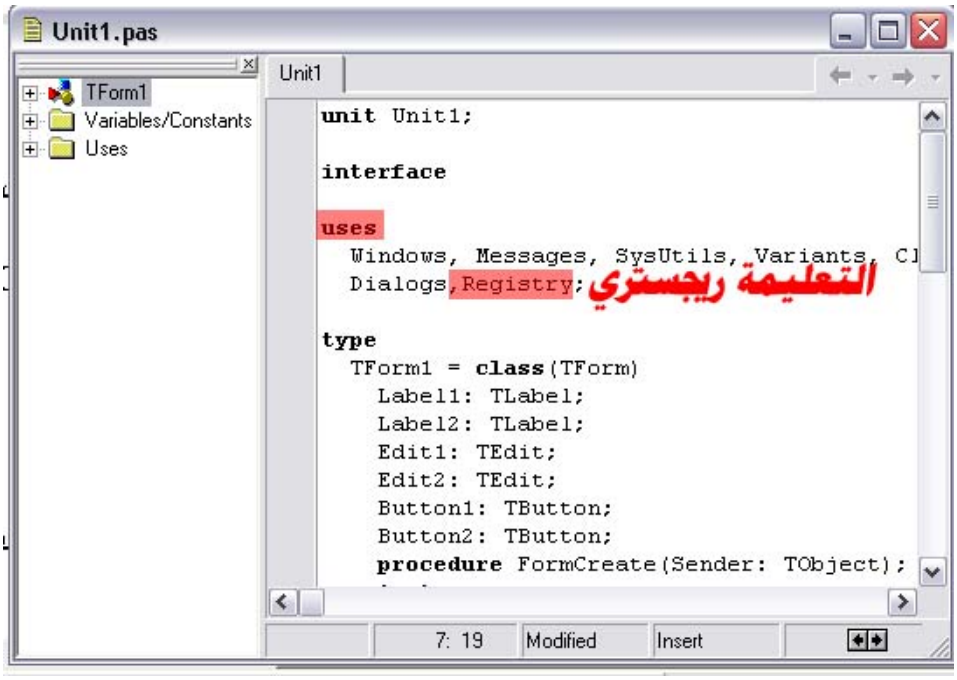
الميزة العظمى في Delphi هي أن نشعرها بأننا نريد استخدام عنصر ما، مثلاً لاستخدام الريجستري يجب أن نخبر Delphi بأننا نريد استخدام الريجستري وهكذا بالنسبة للأعمال البقية وبهذه الطرق نكون قد خفّفنا من ثقل البرنامج.

أما الآن... لإخبار دلفي أننا نريد استخدام الريجستري اتّبع الطرق التالية:

١. افتح مشروع جديد من خلال New>Application.

٢. أنقر نقراً مزدوجاً على النافذة (Form) لإظهار مستكشف الكود (Code Explorer).

٣. أكتب التعليمة Registry في القسم uses كما في الشكل التالي:



الآن أخبرنا Delphi بأننا نريد استخدام الريجستري في برنامجنا لننتقل إلى الخطوة التالية.

تعيين المفاتيح (فتح مفاتيح)

كما ذكرنا فإنَّ المفاتيح هي من أساس الريجستري وتنقسم المفاتيح إلى قسمين:

- ❖ مفاتيح أساسية: وقد مثّلنا لها بـ HKEY_CLASSIC_ROOT وقلنا أنَّ المفتاح الأساسي المذكور يكوّن فيه جميع لواحق الملفات.
- ❖ مفاتيح فرعية: وكما ذكرنا فالمفاتيح الفرعية تتفرّع من المفاتيح الرئيسية ويمكن وجود أكثر من مفتاح فرعي لكل مفتاح.

أما الآن نأتي لطريقة فتح مفتاح في الريجستري من خلال بيئة Delphi وبعد أن نفتح المفتاح وتعيينه يصبح بإمكاننا التعامل مع هذا المفتاح وكتابة قيم ما بداخله.

لفتح مفتاح ما نقوم بما يلي:

١. أضف زر Button على النافذة.

٢. أنقر نقراً مزدوجاً على الزر وأكتب الكود التالي (مع ملاحظة عدم كتابة الأرقام وقد كتبتها لتوضيح الأكواد فقط):

```
1 var
2 Reg: Tregistry;
3 begin
4 Reg:= TRegistry.Create;
5 Reg.RootKey:= HKEY_CURRENT_USER;
6 try
7 Reg.OpenKey('Software\Microsoft',False);
8 except;
9 end;
```

لنأتي إلى شرح هذا الكود بالتفصيل:

- ❖ السطر ١ – ٢: قمنا بتعريف متغير تحت اسم Reg من نوع Tregistry حيث يصبح بإمكاننا الوصول إلى الريجستري من خلال هذا المتغير والذي يحتفظ بكل عمليات الريجستري من فتح مفاتيح، كتابة قيم وغيرها.
- ❖ السطر ٤: قمنا في هذا السطر بخلق الريجستري وهي طريقة تستخدم بكثرة مع الكائنات، يعني يجب عليك خلق الشيء حتى تستفيد منه فهنا قمنا بخلق الريجستري وتعيينه ومن ثم نستخدمه.
- ❖ السطر ٥: قمنا من خلال هذا السطر بفتح المفتاح الرئيسي الذي نريد استخدامه، يمكنك بعد كتابة المتغير بهذا الشكل Reg.RootKey أن

تضغط Ctrl+Space لتحصل على جميع المفاتيح الأساسية، اختر واحد منها واضغط مفتاح Enter.

- السطر ٦: استخدمنا فيه بلوك المقابلة مع الاستثناء (except) أو بعبارة أحرى مع الخطأ الذي يحدث فإن لم تدرج هذه التعليمة فسيحدث خطأ عند تشغيل البرنامج.
- السطر ٧: قمنا من خلال هذا السطر بفتح المفتاح الفرعي الذي نريد استخدامه، يمكنك كتابة المسار بأكمله وكما هو مبين فقد كتبت مسار Software\Microsoft للوصول إلى مفتاح Microsoft، يعني مع ملاحظة أن تكتب جميع المسار.
- السطر ٨-٩: نهاية بلوك الاستثناء.

الآن قمنا بفتح المفاتيح والوصول إليها، وتبقى طريقة كتابة قيم.

كتابة قيم

لكتابة قيمة أو قيم نقوم بالخطوات البسيطة التالية مع ملاحظة أنك قد قمت بالخطوات السابقة وأتممتها بكل صحة، كما لا يخفى عليك أن تضيف Edit إلى النافذة:

١. أضف هذه الجملة بعد السطر رقم ٧ مباشرةً:

```
Reg.writeString('Test',Edit1.Text);
```

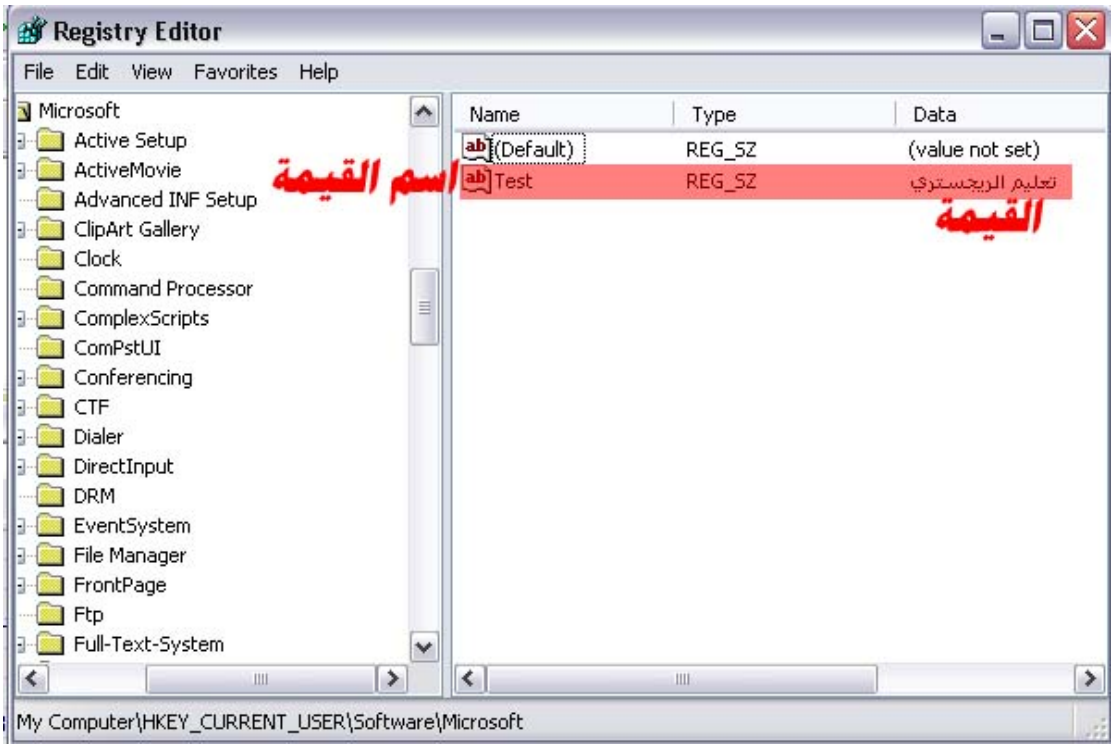
تقوم هذه الجملة بتخزين محتويات الـ Edit تحت قيمة باسم Test، الآن نفّذ البرنامج واضغط على الزر Button وبعد الضغط عليه انتقل إلى الريجستري واذهب إلى المفتاح المذكور لترى أن كل ما كتبته في الـ Edit قد ظهر تحت قيمة باسم Test وفيها جميع محتويات الـ Edit، ويُمكنك استخدام الأنواع التالية أيضاً:

- WriteString: لكتابة قيم نصية.
- WriteCurrency: لكتابة قيم كارانسي.
- WriteBinaryData: لكتابة قيم ثنائية Binary.
- WriteBool: لكتابة قيم الـ Boolean وهي تأخذ True وFalse.
- WriteDate: لكتابة قيم التاريخ.

- ❖ WriteDateTime : لكتابة قيم التاريخ و الوقت .
- ❖ WriteFloat : لكتابة قيم Float .
- ❖ WriteInteger : لكتابة قيم Integer (الأعداد الصحيحة) .
- ❖ WriteExpandString : لكتابة قيم سلسلة موسّعة .
- ❖ WriteTime : لكتابة الوقت .

يمكنك كتابة هذه القيم لإنجاز ما تريد من مهام .

ولرؤية الناتج انتقل إلى مسار المفتاح في الريجستري كما هو موضّح في الصورة:



الأوامر المتقدّمة

هناك سلسلة من الأوامر المتقدّمة أشرحها بطريقة مفصّلة:

- ❖ ValueExists(): أمر للتحقق من أنّ هل القيمة الفلانية موجودة ضمن المفتاح أو لا ، مثال:

```
if Reg.ValueExists(Test) then
   ShowMessage('found...');
else
    ShowMessage('no found...');
```

مع ملاحظة أنك قد قمت سابقاً بفتح المفاتيح المطلوبة لكي تتأكد من وجود القيمة المذكورة.

- ()KeyExists: يستخدم هذا الأمر للتأكد هل أنَّ المفتاح الفلاني موجود أو لا، يعني كالأمر السابق إلا أنه يستخدم للمفاتيح وليس للقيم، مثال:

```
if Reg.KeyExists(Software) then
   ShowMessage('found…);
else
    ShowMessage('no found…);
```

- ()DeleteKey: يستخدم هذا الأمر لحذف مفتاح من الريجستري، مثال:

```
If Reg.DeleteKey('KeyName') then
      الكود الذي تريد تنفيذه عند نجاح الحذف//
else
      الكود الذي تريد تنفيذه عند عدم نجاح الحذف//
```

يقوم الأمر المذكور بحذف المفتاح KeyName من الريجستري، يجب أن تكون حذراً عن استخدامك لهذا الأمر ولا تحذف المفاتيح الضرورية أو التي لا تعرف ما هي إلا أن تتأكد منها.

- ()Reg.DeleteValue: كالأمر السابق تماماً إلا أنه يحذف قيمة بدلاً من مفتاح.
- ReadString: إذا أردنا أن نقرأ قيمة من الريجستري فنستخدم هذا الأمر كما في المثال التالي:

```
ShowMessage(Reg.ReadString('Test'));
```

يقوم هذا المثال بقراءة القيمة Test من الريجستري وإظهارها بواسطة رسالة، يمكنك استخدام أي كومبوننت آخر لوضع القيم المقرأة بداخله.

- ()Reg.CreateKey: يستخدم هذا الأمر لإنشاء مفتاح جديد، ما عليك سوى كتابة المفتاح وسيتم إنشاءه، مثال:

```
Reg.CreateKey('KeyName');
```

سيتم إنشاء المفتاح KeyName تحت المفتاح المفتوح مسبقاً، وإذا لم تفتح أي مفتاح مسبقاً فسيتم إنشاءه تحت المفتاح الرئيسي.

- ❖ ()MoveKey: يستخدم هذا الأمر لنقل أو نسخ مفتاح مع قيمه إلى مكان تحدده أنت، مثال:

```
Reg.MoveKey('Ahmad','ali',false);
```

سيتم نسخ المفتاح Ahmad إلى مفتاح جديد باسم ali، والبارامتر الأخير يؤكد ما إذا أردت حذف المفتاح Ahmad أو لا، ضع False لكي لا تحذف، ضع True لكي تحذفه.

- ❖ ()CurrentPath: يجلب هذا الأمر المفتاح المفتوح حالياً، مثال:

```
ShowMessage(Reg.CurrentPath);
```

- ❖ ()Reg.OpenKeyReadOnly: يستخدم هذا الأمر لفتح مفتاح للقراءة فقط.
- ❖ أوامر Read: وهي أوامر عديدة مثل ReadString وغيرها... تستخدم هذه الأوامر لقراءة قيمة من الريجستري.
- ❖ RenameValue: يستخدم هذا الأمر لاستبدال اسم قيمة معينة، مثال:

```
Reg.RenameValue('Test','myValue');
```

سوف يستبدل اسم القيمة Test إلى myValue.

وهناك أوامر عديدة في الريجستري اكتشفها بنفسك.

أما الآن فنأتي إلى إنشاء برنامج Background وهو برنامج رائع رغم صغر حجمه.

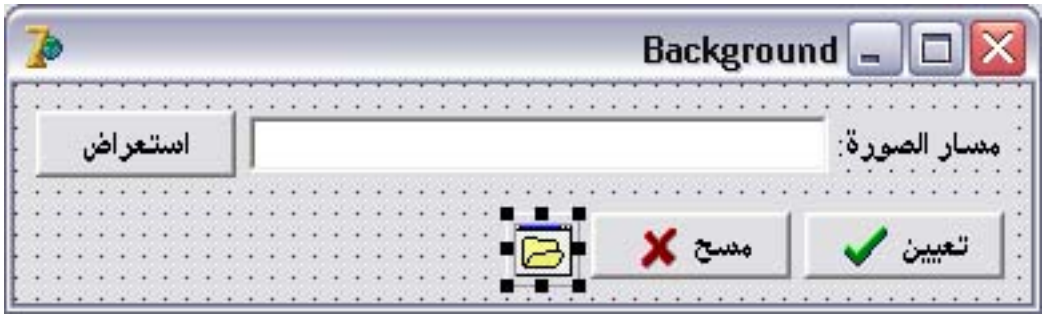
برنامج Background

لا يخفى عليكم أن للريجستري أوامراً خاصة تحسّن من كيفية وأداء الويندوز وهذه الأوامر تجدونها موجودة في كتب مستقلّة وللأسف فالكتب العربية قليلة جداً في هذا المجال لكنني بإذن الله سأترجم كتاب بعنوان (الريجستري ٢٥٠ أمر محافظ)

وهو كتاب رغم صغر حجمه والذي لا يتعدى الـ ٣٠٠ صفحة إلا أنها يعتبر كتاباً فعالاً لتعلّم الأوامر الخفية في الريجستري والتي تحسّن من أداء الويندوز.

سأستخدم في هذا المثال العملي قيمة باسم BackBitmapShell وهي قيمة خاصة بالريجستري تمكّنك من إنشاء صورة ووضعها كلخفية لشريط الأدوات.

قم بتصميم الشكل التالي:



البرنامج سيكون بالشكل التالي: يضغط المستخدم على زر "استعراض" ويختار الصورة التي يريدها ثم يضغط على زر "تعيين" لتعيينها كلخفية لشريط الأدوات في المستكشف وجهاز الكمبيوتر، وإذا أراد المستخدم أن يلغي الصورة فيضغط على زر "مسح"، لاحظ أنه يجب عليك أن تدرج اللاحقة bmp واللاحقة gif إلى مربع حوار فتح OpenDialog.

بناء البرنامج

اتبع الخطوات التالية لبناء البرنامج:

١. ابدأ تطبيقاً جديداً.

٢. أدرج التعليمة Registry في القسم uses كما وضحت سابقاً.

٣. أنقر نقراً مزدوجاً على النافذة وأكتب الكود التالي:

```
var
RegEx: Tregistry;
begin
RegEx:= TRegistry.Create;
RegEx.RootKey:= HKEY_CURRENT_USER;
try
RegEx.OpenKey('Software\Microsoft\Internet
Explorer\Toolbar',True);
 Edit1.Text:= RegEx.ReadString('BackBitmapShell');
 except
 RegEx.Free;
```

```
 End;
```

٤. أنقر نقراً مزدوجاً على زر "تعيين" وأكتب الكود التالي:

```
var
Reg: Tregistry;
begin
Reg:= TRegistry.Create;
Reg.RootKey:= HKEY_CURRENT_USER;
try
Reg.OpenKey('Software\Microsoft\Internet
Explorer\Toolbar',True);
Reg.WriteString('BackBitmapShell',Edit1.Text);
ShowMessage('تم تعيين الخلفية بنجاح');
except
Reg.Free;
End;
```

٥. أنقر نقراً مزدوجاً على زر "مسح" وأكتب الكود التالي:

```
var
Reg: Tregistry;
begin
Reg:= TRegistry.Create;
Reg.RootKey:= HKEY_CURRENT_USER;
try
Reg.OpenKey('Software\Microsoft\Internet
Explorer\Toolbar',True) ;
If MessageDlg،'هل تريد مسح الخلفية؟'(mtInformation,[mbYes,mbNo],0) =
mrYes then
begin
 if Reg.DeleteValue('BackBitmapShell') then
begin
ShowMessage('تم مسح الخلفية بنجاح');
end else begin
ShowMessage('هناك خطأ في المسح!');
End;
End;
except
Reg.Free;
End;
```

٦. اضغط على F9 لتشغيل البرنامج، واختر صورة بامتداد bmp أو gif لكي تصبح خلفية لشريط الأدوات.

الآن... والحمد لله انتهينا من هذا البرنامج البسيط والفعّال في نفس الوقت.

قد تكون الأكواد التي كتبتها مبعثرة بعض الشيء إلا أنني تفاديت هذه المشكلة بإرفاق البرنامج نفسه، لقد أرفقت لكم البرنامج لكي تطلعوا على أكواده.

هذا هو كل ما في الريجستري، أنظروا كيف أنَّ Delphi تسهّل الأشياء الصعبة، ففي Visual Basic يجب عليك كتابة طن من الأكواد لكي تحصل على هذا البرنامج.

أوصيكم بأن تتمرسوا ولو لساعةٍ واحدة على أوامر الريجستري وتكتبوا برامجاً من أنفسكم.

ترقّبوا الكتب / الكتيبات التالية:

- الريجستري ٢٥٠ أمر محافظ.
- شرح للـ Cpanel الإصدار ٩ لسيرفر Linux.
- فكرة برمجة أنظمة التشغيل OS.
- C++ في يومٍ واحدٍ فقط!.
- والمزيد من الكُتب الأخرى...

والسلام عليكم ورحمة الله وبركاته

أحمد المياحي

العراق

يوم الإثنين ٥/٤/٢٠٠٤م