IniEdit is a control to ease the burden of reading and writing to INI files.

IniEdit will be placed on the 'Data Access' tab of Delphi. Of course, if you don't like this or anything else about IniEdit, you can change it to suit your preferences. The source code is in EasyINI.pas.

IniEdit only has two unique properties. These you can set at design time and if necessary modify at run time.

The two properties are:

FileLocation property - This can be set to one of three settings. **'ApplicationPath'** will put the INI file in the same directory as your executable. This is the default. **'PathInFileName'** means you are putting the full path and file name in the 'FileName' property. **'WindowsPath'** puts the INI file in the Windows directory.

FileName property - This is the name of the INI file. If you set the FileLocation property to 'PathInFileName' then you can put the full path and file name in this property. If you leave off the file extension, then '.INI' is added to the file name. There is no default value for 'FileName'. I suppose it could have been WIN.INI, but I didn't want to make it too easy to write to that already bloated file.

The following procedures write info to the file you specified in the two properties above. You simple specify the **Section**, **Entry** and **Value** you want to write. If you are not familiar with this, search Delphi's Windows API help for 'WritePrivateProfileString'.

procedure **WriteStr**(Section, Entry, Value: string);

procedure **WriteBool**(Section, Entry: string; Value: boolean);

procedure **WriteChar**(Section, Entry: string; Value: char);

procedure **WriteInt**(Section, Entry: string; Value: LongInt);

procedure **WriteReal**(Section, Entry: string; Value: Extended);

Sample code:

```
INI.Filename := 'AGES.INI';
INI.FileLocation := ApplicationPath;
INI.WriteInt ('Numbers',  'MyAge', 12);
```

The resulting AGES.INI file would look like this:

```
[Numbers]
MyAge=12
```

The read functions simply parallel the write functions so you can read in what you have stored. The default value is returned if the read fails to find the data.

function **ReadStr**(Section, Entry, DefaultValue: string): string;

function **ReadBool**(Section, Entry: string; DefaultValue: boolean): boolean;

function **ReadChar**(Section, Entry: string; DefaultValue: char): char;

function **ReadInt**(Section, Entry: string; DefaultValue: LongInt): LongInt;

function **ReadReal**(Section, Entry: string; DefaultValue: Extended): Extended;

Sample code:

```
var HowOld: integer;

INI.Filename := 'AGES.INI';
INI.FileLocation := ApplicationPath;
HowOld := INI.ReadInt ('Numbers',  'MyAge', 99);
```

If the MyAge entry is found and there's no conversion error then HowOld will be set to 12; otherwise, HowOld is set to the default value 99.

EraseEntry erases the Entry specified under a Section specified. This means the entry is gone, not just the right side of = is blank.

procedure **EraseEntry**(Section, Entry: string);

Sample code:

```
INI.Filename := 'AGES.INI';
INI.FileLocation := ApplicationPath;
INI.EraseEntry('Numbers',  'MyAge');
```

Now AGES.INI would look like this:

```
[Numbers]
```

EraseSection erases a whole Section. The section name and all Entries that belong to it are erased.

procedure **EraseSection**(Section: string);

Sample code:

```
INI.Filename := 'AGES.INI';
INI.FileLocation := ApplicationPath;
INI.EraseSection('Numbers',  'MyAge');
```

AGES.INI would now be an empty file since [Numbers] and any remaining entries under [Numbers] would be erased.

I would suggest puting EasyINI.dcu and EasyINI.dcr files in your \DELPHI\LIB directory and then follow the instructions for installing VCL's.

I hope this makes life easier for you. **Its free and you got the source code so don't hold me responsible for anything, Okay?**