

Forum:[Hack Forums](#)

Thread: [.INI File Reader/Writer](#)

Velocity™ - May 22, 2015, 9:37 pm

I just made a very simple .ini file reader/writer for newbies who are interested into this idea.

It can be used for many things such as:

- Settings Save
- Saving strings

The Program(Mediafire):

http://www.mediafire.com/download/3fput1vfyitzarr/INI_Reader,_Writer.exe

The Program Virus Total:

<https://www.virustotal.com/en/file/e48962a19ed031e0055608a5ca7fccca66a83b68392baba17b43a546d7a0d3c0/analysis/1432355585/> (its 10kb so make of that as you will)

The Source: <http://www.mediafire.com/download/blbrtuee7t2t1dl/WindowsFormsApplication1.rar>

Thanks!

SL x TnT - May 23, 2015, 1:04 am

Your program download / source links are broken right now. Only really posting to remind myself to look at the source later. That one detection is making me curious to see your source.

Velocity™ - May 23, 2015, 1:14 am

Updating now with the correct links, apologies!

CodeAssassin - May 23, 2015, 1:33 am

Had a look, This is a lot better coded then your last app. You have a few tiny things i would change but there more personal preferences more then anything. Also you left an empty label sub in. But na looks good. Simple but effective for the newbies :)

Velocity™ - May 23, 2015, 1:41 am

(May 23, 2015 01:33 AM)sweetysuckscock Wrote: [\[CITE\]](#) Had a look, This is a lot better coded then your last app. You have a few tiny things i would change but there more personal preferences more then anything. Also you left an empty label sub in. But na looks good. Simple but effective for the newbies :)

Thank you brother :)

Appreciate it!

Serux - May 23, 2015, 11:25 am

A few things

Code public string FilePath

```
{  
    get { return this.filePath; }  
    set { this.filePath = value; }  
}
```

Can be simplified to

Code public string FilePath { get; set; }

A few naming convention issues, but that's not a big deal.
Also please learn to use var. It's so much cleaner!

But you do realize you don't have to import functions from kernel.dll to read/write, right?

Your class1.cs should really be in your WindowsFormsApplication1 namespace as well.

You're not in java. this.variable name is implied. Example: this.FilePath = filePath; will be the same result as FilePath = filePath;

Might I suggest also having class1 as an object? Rather than using static read/write methods use some OOP!

Nikey646 - May 23, 2015, 12:18 pm

(May 23, 2015 11:25 AM)Serux Wrote: [\[CITE\]](#) A few things

```
Code public string FilePath
{
    get { return this.filePath; }
    set { this.filePath = value; }
}
```

Can be simplified to

```
Code public string FilePath { get; set; }
```

A few naming convention issues, but that's not a big deal.
Also please learn to use var. It's so much cleaner!

But you do realize you don't have to import functions from kernel.dll to read/write, right?

Your class1.cs should really be in your WindowsFormsApplication1 namespace as well.

You're not in java. this.variable name is implied. Example: this.FilePath = filePath; will be the same result as FilePath = filePath;

Might I suggest also having class1 as an object? Rather than using static read/write methods use some OOP!

In the case of the first constructor, he is actually referencing the private variable "filePath", but the variable passed to the constructor is of the same name, so this.* is required.

@OP: Why is there a p object, and constructor to set the p object? I don't see it being used anywhere.

Serux - May 23, 2015, 12:41 pm

(May 23, 2015 12:18 PM)Nikey646 Wrote: [\[CITE\]](#) In the case of the first constructor, he is actually referencing the private variable "filePath", but the variable passed to the constructor is of the same name, so this.* is required.

@OP: Why is there a p object, and constructor to set the p object? I don't see it being used anywhere.

I'm saying he could get rid of filePath completely and just have FilePath as an encapsulated variable. I see no real use of filePath without using Class1 as an object.

SL x TnT - May 23, 2015, 10:21 pm

(May 23, 2015 11:25 AM)Serux Wrote: [\[CITE\]](#) Also please learn to use var. It's so much cleaner!

var is fine, if used "correctly."

Use implicit typing for local variables when the type of the variable is obvious from the right side of the assignment, or when the precise type is not important.

<https://msdn.microsoft.com/en-us/library/ff926074.aspx>

I've seen some people only use var. It makes their code difficult to read. For me, I hate using var outside of testing something quickly. It's just a personal preference knowing what a variable is before having to read the entire line.

if not true then false - May 24, 2015, 5:15 am

(May 23, 2015 10:21 PM)SL x TnT Wrote: [\[CITE\]](#) I've seen some people only use var. It makes their code difficult to read. For me, I hate using var outside of testing something quickly. It's just a personal preference knowing what a variable is before having to read the entire line.

I've never saw somebody only use var (God fucking help the person that tries to read their code) but I'm on the same side as you, it's helpful to know what type a variable is against having to look at the right side.

Even when I could use var, I wouldn't; it just doesn't look as clean to me.

0xDEAD10CC - May 24, 2015, 5:22 pm

(May 23, 2015 11:25 AM)Serux Wrote: [\[CITE\]](#) A few things

Code public string FilePath

```
{
    get { return this.filePath; }
    set { this.filePath = value; }
}
```

Can be simplified to

Code public string FilePath { get; set; }

A few naming convention issues, but thats not a big deal.

Also please learn to use var. Its so much cleaner!

But you do realize you dont have to import functions from kernel dll to read/write, right?

Your class1.cs should really be in your WindowsFormsApplication1 namespace as well.

You're not in java. this.variable name is implied. Example: this.FilePath = filePath; will be the same result as FilePath = filePath;

Might I suggest also having class1 as an object? Rather than using static read/write methods use some OOP!

He could simplify it to that, but the bigger problem is that it's publicly settable, when it probably should NOT be. It should be set from the constructor and the Filepath should remain that way IMHO. There's no real reason for it to change, but you should allow the getter to still be public so that the programmer can still access that filepath string when they need it.

(May 24, 2015 05:15 AM)if not true then false Wrote: [\[CITE\]](#) I've never saw somebody only use var (God fucking help the person that tries to read their code) but I'm on the same side as you, it's helpful to know what type a variable is against having to look at the right side.

Even when I could use var, I wouldn't; it just doesn't look as clean to me.

You would rather still have a bunch of enumerable types and generic parameters visible in your

code when it really doesn't have to be in the case of LINQ usage? In most cases the context of the code and the names if you have good naming conventions are self-documenting enough, and if they aren't, they should be, which means you have an issue elsewhere, aside from whether or not to use the var keyword...

Additionally, what about for anonymous types?

```
Code var obj = new { X = 10, Y = 20 };
```

if not true then false - May 26, 2015, 2:30 am

(May 24, 2015 05:22 PM)0xDEAD10CC Wrote: [\[CITE\]](#) You would rather still have a bunch of enumerable types and generic parameters visible in your code when it really doesn't have to be in the case of LINQ usage? In most cases the context of the code and the names if you have good naming conventions are self-documenting enough, and if they aren't, they should be, which means you have an issue elsewhere, aside from whether or not to use the var keyword...

Additionally, what about for anonymous types?

```
Code var obj = new { X = 10, Y = 20 };
```

Remind me how I should follow your exact way of programming. I like to have types clearly shown because it makes it look cleaner and it makes it easier to read for me. As for anonymous types, I've never had to use them so frankly I don't care about them. When and if I ever do need to use them, obviously I'll use var but I would rather use var as little as possible except for cases of convenience or requirement.

KingCobraXD - May 26, 2015, 7:30 pm

(May 24, 2015 05:22 PM)0xDEAD10CC Wrote: [\[CITE\]](#) He could simplify it to that, but the bigger problem is that it's publicly settable, when it probably should NOT be. It should be set from the constructor and the FilePath should remain that way IMHO. There's no real reason for it to change, but you should allow the getter to still be public so that the programmer can still access that filepath string when they need it.

Then you can instead do:

```
Code public string FilePath { get; private set; }
```

CodeAssassin - May 27, 2015, 9:33 am

(May 26, 2015 07:30 PM)KingCobraXD Wrote: [\[CITE\]](#) Then you can instead do:

```
Code public string FilePath { get; private set; }
```

I learnt this yesterday specifically for passing arguments in a delegate. Where you only want that argument to be read only. Good tip that :)