

Forum:[Hack Forums](#)

Thread: [Very Simple I/O *.INI Reader](#)

Tedd - January 11, 2014, 2:59 am

Hey, for an application I am making, I need to be able to read an INI file. So I decided to write up a very basic I/O StreamReader ini file reader.

Example usage:

Added a little class so you can grab a section and not have to use .GetValue("name", "section") each time, which would require it to open a stream reader each time.

Code:

Code Imports System.IO

Public Class INIReader

Public ReadOnly file As String

Sub New(ByVal inifile As String)

If IO.File.Exists(inifile) Then

file = inifile

Else

'If our file isn't found, we want to let the user know.

Throw New FileNotFoundException("File does not exist.", inifile)

End If

End Sub

Public Function GetValue(ByVal value As String, Optional ByVal section As String = Nothing) As String

Using reader As New IO.StreamReader(file)

Dim temp As String

If section Is Nothing Then

While Not reader.EndOfStream

temp = reader.ReadLine().Trim

'If the line is null then we don't need to read it

If temp Is Nothing Then Continue While

'If the line is blank then we don't need to read it

If temp.Length = 0 Then Continue While

'If the line is a comment then we don't need to read it

If temp(0) = ";" Then Continue While

'If the line is a section, then we don't need to read it

If temp(0) = "[" Then Continue While

```
'If the line begins with the property name, then we return everything after the "=".
If temp.Substring(0, temp.IndexOf("="c)).Trim = value Then
Return temp.Substring(temp.IndexOf("="c) + 1).Trim
End If
End While
```

```
'Either we got to the next section or the end of the file and nothing was found.
Return Nothing
Else
While Not reader.EndOfStream
temp = reader.ReadLine.Trim
```

```
'If the line is null then we don't need to read it
If temp Is Nothing Then Continue While
```

```
'If the line is blank then we don't need to read it
If temp.Length = 0 Then Continue While
```

```
'If the line is a comment then we don't need to read it
If temp(0) = ";"c Then Continue While
```

```
If temp.Substring(0, section.Length + 2) = "[" & section & "]" Then
'We've found the section, we need to clear the temp variable to use it for our property now
temp = Nothing
```

```
While Not reader.EndOfStream
temp = reader.ReadLine().Trim
```

```
'If the line is null then we don't need to read it
If temp Is Nothing Then Continue While
```

```
'If the line is blank then we don't need to read it
If temp.Length = 0 Then Continue While
```

```
'If the line is a comment then we don't need to read it
If temp(0) = ";"c Then Continue While
```

```
'If the line is a section, then the section we were looking for didn't have our property, so we exit.
If temp(0) = "["c Then Exit While
```

```
'If the line begins with the property name, then we return everything after the "=".
If temp.Substring(0, temp.IndexOf("="c)).Trim = value Then
Return temp.Substring(temp.IndexOf("="c) + 1).Trim
End If
End While
End If
End While
```

```
'Either we got to the next section or the end of the file and nothing was found.
Return Nothing
End If
```

temp = Nothing

reader.Close()
End Using
End Function

Public Function GetProperties(ByVal section As String) As List(Of String())
GetProperties = New List(Of String())

Using reader As New IO.StreamReader(file)
Dim temp As String

While Not reader.EndOfStream
temp = reader.ReadLine.Trim

'If the line is null then we don't need to read it
If temp Is Nothing Then Continue While

'If the line is blank then we don't need to read it
If temp.Length = 0 Then Continue While

'If the line is a comment then we don't need to read it
If temp(0) = ";"c Then Continue While

'If the string isn't even as long as "[" & section & "]" it's obviously not the section
If temp.Length < section.Length + 2 Then Continue While

If temp.Substring(0, section.Length + 2) = "[" & section & "]" Then
'We've found the section, we need to clear the temp variable to use it for our property now
temp = Nothing

While Not reader.EndOfStream
temp = reader.ReadLine().Trim

'If the line is null then we don't need to read it
If temp Is Nothing Then Continue While

'If the line is blank then we don't need to read it
If temp.Length = 0 Then Continue While

'If the line is a comment then we don't need to read it
If temp(0) = ";"c Then Continue While

'If the line is a section, then the section we were looking for didn't have our property, so we exit.
If temp(0) = "["c Then Exit While

'If the line begins with the property name, then we return everything after the "=".
If temp.Substring(0, temp.IndexOf("="c)).Trim.Length <> 0 Then
GetProperties.Add({temp.Substring(0, temp.IndexOf("="c)).Trim,
temp.Substring(temp.IndexOf("="c) + 1).Trim})
End If
End While
End If

End While

'Either we got to the next section or the end of the file and nothing was found.

Return GetProperties

temp = Nothing

reader.Close()

End Using

End Function

End Class

Public Class INISection

Public properties As List(Of String())

Sub New(ByVal reader As INIReader, ByVal section As String)

properties = reader.GetProperties(section)

End Sub

Public Function GetValue(ByVal name As String)

Return properties.Find(Function(x) x(0) = name)(1)

End Function

End Class

Thank you to [Lodestone™](#) for a C# version:

Code using Microsoft.VisualBasic;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Diagnostics;

using System.IO;

public class IniReader

{

public readonly string file;

public IniReader(string inifile)

{

if (File.Exists(inifile))

{

file = inifile;

}

else

{

throw new FileNotFoundException("File does not exist.", inifile);

}

}

public string GetValue(string value, string section = null)

{

using (StreamReader reader = new StreamReader(file))

{

```

string temp = null;

if (section == null)
{
    while (!reader.EndOfStream)
    {
        temp = reader.ReadLine().Trim();
        if (temp == null)
            continue;
        if (temp.Length == 0)
            continue;
        if (temp[0] == ';')
            continue;
        if (temp[0] == '[')
            continue;
        if (temp.Substring(0, temp.IndexOf('=')).Trim() == value)
        {
            return temp.Substring(temp.IndexOf('=') + 1).Trim();
        }
    }
    return null;
}
else
{
    while (!reader.EndOfStream)
    {
        temp = reader.ReadLine().Trim();
        if (temp == null)
            continue;
        if (temp.Length == 0)
            continue;
        if (temp[0] == ';')
            continue;
        if (temp.Substring(0, section.Length + 2) == "[" + section + "]")
        {
            temp = null;

            while (!reader.EndOfStream)
            {
                temp = reader.ReadLine().Trim();
                if (temp == null)
                    continue;
                if (temp.Length == 0)
                    continue;
                if (temp[0] == ';')
                    continue;
                if (temp[0] == '[')
                    break;
                if (temp.Substring(0, temp.IndexOf('=')).Trim() == value)
                {
                    return temp.Substring(temp.IndexOf('=') + 1).Trim();
                }
            }
        }
    }
}

```

```

    }
    }

    return null;
}

temp = null;

reader.Close();
}
}

public List<string[]> GetProperties(string section)
{
    List<string[]> functionReturnValue = default(List<string[]>);
    functionReturnValue = new List<string[]>();

    using (StreamReader reader = new StreamReader(file)) {
        string temp = null;

        while (!reader.EndOfStream) {
            temp = reader.ReadLine().Trim();
            if (temp == null)
                continue;
            if (temp.Length == 0)
                continue;
            if (temp[0] == ';')
                continue;
            if (temp.Length < section.Length + 2)
                continue;

            if (temp.Substring(0, section.Length + 2) == "[" + section + "]") {
                temp = null;

                while (!reader.EndOfStream) {
                    temp = reader.ReadLine().Trim();
                    if (temp == null)
                        continue;
                    if (temp.Length == 0)
                        continue;
                    if (temp[0] == ';')
                        continue;
                    if (temp[0] == '[')
                        break;
                    if (temp.Substring(0, temp.IndexOf('=')).Trim().Length != 0) {
                        functionReturnValue.Add(new string[] {temp.Substring(0, temp.IndexOf('=')).Trim(),
temp.Substring(temp.IndexOf('=') + 1).Trim() });
                    }
                }
            }
        }
    }
    return functionReturnValue;
}

```

```

        temp = null;

        reader.Close();
    }
    return functionReturnValue;
}

}

public class IniSection
{

    public List<string[]> properties;
    public IniSection(IniReader reader, string section)
    {
        properties = reader.GetProperties(section);
    }

    public object GetValue(string name)
    {
        return properties.Find(x => x[0] == name)[1];
    }
}

```

If anyone has any suggestions on things I overlooked, please post. I tried to cover all possible scenarios.

GammaRay - January 11, 2014, 3:15 am

Vouch for this method. Gets me high.

Just kidding. But thanks for this. Well commented.

Anon?M ID - January 11, 2014, 4:30 am

nice code you got there. but i have simplest module for ini reader

```

Code Imports System
Imports System.IO
Imports System.Runtime.InteropServices
Imports System.Text
Namespace IniRW
    Public Class IniFile
        Public path As String
        <DllImport("kernel32")> _
        Private Shared Function WritePrivateProfileString(ByVal section As String, ByVal key As String,
ByVal val As String, ByVal filePath As String) As Long
        End Function
        <DllImport("kernel32")> _
        Private Shared Function GetPrivateProfileString(ByVal section As String, ByVal key As String,
ByVal def As String, ByVal retVal As StringBuilder, ByVal size As Integer, ByVal filePath As String)
As Integer
        End Function
        Public Sub New()
            path = Environment.CurrentDirectory + "\Setting.ini"
        End Sub
    End Class
End Namespace

```

```

Public Sub IniWriteValue(ByVal Section As String, ByVal Key As String, ByVal Value As String)
    WritePrivateProfileString(Section, Key, Value, Me.path)
End Sub
Public Function IniReadValue(ByVal Section As String, ByVal Key As String) As String
    Dim temp As New StringBuilder(255)
    Dim i As Integer = GetPrivateProfileString(Section, Key, "", temp, 255, Me.path)
    Return temp.ToString()
End Function
End Class
End Namespace

```

zyvox - January 11, 2014, 5:12 am

Looks nice, also well commented. Nice work :)

Tedd - January 11, 2014, 5:35 am

(Jan 11, 2014 06:12 AM)Z??o?™ Wrote: [\[CITE\]](#) Looks nice, also well commented. Nice work :) Thanks! It's ridiculously simple concept, but it works with just about anything you can throw at it. I think I benchmarked it at 10 thousand .GetValues() in 1.7 seconds.

Hoping that by commenting, people can understand what I'm trying to do better and if they read what it's trying to do and realize "oh, that can be done a ton easier this way", they will post here.

ident - January 11, 2014, 11:17 am

The last thread you started method scope variables using pascal case. Now you are using camel case after your hissy fit about no standards. If you cant even follow your own you really are a joke. Oh ever heard of path.combine?

Zettabit - January 11, 2014, 1:05 pm

Seems nice. I have one that I found that I've been using and it basically works with the same principles. You should use properties for the sections though, that way you can search the array for the section, and create a variable that can access that specific section only.

For example, here's a .ini file:

```

Code [program]
width=50
height=150

```

```

[settings]
color=red
border=black

```

An example of what I mean, is like this:

```

Code Dim programini As INIReader.Section = INIReader.GetSection("program")
MessageBox.Show(programini.GetValue("width"))

```

This would be useful if you have large amounts of values in one section, and don't want to keep using .GetValue("section", "property").

Tedd - January 11, 2014, 3:06 pm

(Jan 11, 2014 12:17 PM)ident Wrote: [\[CITE\]](#) The last thread you started method scope variables using pascal case. Now you are using camel case after your hissy fit about no standards. If you cant even follow your own you really are a joke. Oh ever heard of path.combine?

Oh the horrors, I'm such a shitty programmer because I didn't capitalize my variable right. Oh god life must be so hard for you that you are forced to read a lowercase letter verses an uppercased.

Yes I have heard of path.combine, truth is it doesn't matter which one you use, because they both execute about the same speed. If you want to get technical, using simple string manipulation is actually faster than calling Path.Combine()

It's like you to argue about something that small, it's almost like arguing about the capitalization of a name which has no relevance at runtime. :O

ident - January 11, 2014, 3:13 pm

why oh why are you even doing a test with path.combine....When the hell would you ever loop combine, you make this forum a fucking joke. There is nothing to test. To combine paths, and wait for it the whole point of the framework is to use it classes. To combine paths... guess what you should be using..Jesus

Tedd - January 11, 2014, 3:21 pm

(Jan 11, 2014 04:13 PM)ident Wrote: [\[CITE\]](#) why oh why are you even doing a test with path.combine....When the hell would you ever loop combine, you make this forum a fucking joke. There is nothing to test. To combine paths, and wait for it the whole point of the framework is to use it classes. To combine paths... guess what you should be using..Jesus

It's a loop because it shows the speed if you were using the function hundreds of times, do you understand logic?

Why would you use the framework's function, if it's slower than the function you use?? Riddle me that.

(Jan 11, 2014 02:05 PM)Crack Dealer Wrote: [\[CITE\]](#) Seems nice. I have one that I found that I've been using and it basically works with the same principles. You should use properties for the sections though, that way you can search the array for the section, and create a variable that can access that specific section only.

For example, here's a .ini file:

Code [program]

width=50

height=150

[settings]

color=red

border=black

An example of what I mean, is like this:

Code Dim programini As INIReader.Section = INIReader.GetSection("program")

MessageBox.Show(programini.GetValue("width"))

This would be useful if you have large amounts of values in one section, and don't want to keep using .GetValue("section", "property").

How's that? It's actually faster because now it doesn't have to read the file each time you want to grab a property.

The new code will be put in the OP after I post this

Zettabit - January 11, 2014, 4:48 pm

(Jan 11, 2014 04:21 PM)Tedd Wrote: [\[CITE\]](#) It's a loop because it shows the speed if you were using

the function hundreds of times, do you understand logic?

Why would you use the framework's function, if it's slower than the function you use?? Riddle me that.

How's that? It's actually faster because now it doesn't have to read the file each time you want to grab a property.

The new code will be put in the OP after I post this
That's actually exactly what I was referring to. Looks nice. :)

If I think of any other suggestions, I'll post them.

DeanReynolds - January 11, 2014, 5:06 pm

(Jan 11, 2014 12:17 PM)ident Wrote: [\[CITE\]](#) The last thread you started method scope variables using pascal case. Now you are using camel case after your hissy fit about no standards. If you cant even follow your own you really are a joke. Oh ever heard of path.combine?

If that's how you are judging peoples code then you really are sad, what are you 8?

I used to start my variables lower case until I started otherwise like:

Code callFunction();Code CallFunction();
On topic: Nice simple function release, thank you.

ident - January 11, 2014, 6:09 pm

(Jan 11, 2014 06:06 PM)DeanReynolds Wrote: [\[CITE\]](#) If that's how you are judging peoples code then you really are sad, what are you 8?

I used to start my variables lower case until I started otherwise like:

Code callFunction();Code CallFunction();
On topic: Nice simple function release, thank you.
Thats because i am a programmer. You are not.

DeanReynolds - January 11, 2014, 6:13 pm

(Jan 11, 2014 07:09 PM)ident Wrote: [\[CITE\]](#) Thats because i am a programmer. You are not.
Oh great, we have another kid here. You think other people aren't programmers because they don't programme with certain casing and exactly how you want it. You are a sad, sad child.

Tedd - January 11, 2014, 6:16 pm

(Jan 11, 2014 07:09 PM)ident Wrote: [\[CITE\]](#) Thats because i am a programmer. You are not.
I like how you didn't find any retarded comeback for my message so you don't reply ;)

shebang - January 12, 2014, 4:42 am

(Jan 11, 2014 04:21 PM)Tedd Wrote: [\[CITE\]](#) It's a loop because it shows the speed if you were using the function hundreds of times, do you understand logic?

Why would you use the framework's function, if it's slower than the function you use?? Riddle me that.

Why would you be running it that many times for it to make enough of a difference? You need to

combine a path once in 99% of the situations it's going to be applied in.

I use the same practices you do, but he makes a good point. Also, one more thing: You don't exactly need to use `Console.Out.WriteLine()`, since it's the same as `Console.WriteLine()`. This is what the latter looks like:

```
Code Public Shared Sub WriteLine(value As String)
    Out.WriteLine(value)
End Sub
```

`Out.WriteLine` should be used when you need to apply a `TextWriter` (i.e. piping other application's output).

Washi - January 12, 2014, 12:51 pm

(Jan 11, 2014 04:21 PM)Tedd Wrote: [\[CITE\]](#) It's a loop because it shows the speed if you were using the function hundreds of times, do you understand logic?

Why would you use the framework's function, if it's slower than the function you use?? Riddle me that.

I believe there are some platform issues when you use the string concatenation instead of the `Path.Combine` method. I can recall having some issues with combining paths on Japanese systems (or maybe another language) which used for example other directory separators. Therefore, you are always save if you use the `Combine` method.

?eumonic - January 12, 2014, 9:57 pm

(Jan 12, 2014 01:51 PM)TheUnknownProgrammer Wrote: [\[CITE\]](#) I believe there are some platform issues when you use the string concatenation instead of the `Path.Combine` method. I can recall having some issues with combining paths on Japanese systems (or maybe another language) which used for example other directory separators. Therefore, you are always save if you use the `Combine` method.

See, THAT is good reasoning. With ident saying that "you should just because you should" because "I am a programmer and you are not," he is not explaining why something should be done a certain way, he is just being a dick.

Oscura - January 12, 2014, 11:38 pm

Thanks for the source. Seems very handy!

nuk - January 13, 2014, 6:29 am

I know a lot of people look for this code & it's one of the things I'm asked about a lot.... I'm gonna send them to your code :) Great share :)

Tedd - January 13, 2014, 9:28 am

(Jan 12, 2014 01:51 PM)TheUnknownProgrammer Wrote: [\[CITE\]](#) I believe there are some platform issues when you use the string concatenation instead of the `Path.Combine` method. I can recall having some issues with combining paths on Japanese systems (or maybe another language) which used for example other directory separators. Therefore, you are always save if you use the `Combine` method.

(Jan 12, 2014 10:57 PM)Lodestone™ Wrote: [\[CITE\]](#) See, THAT is good reasoning. With ident saying that "you should just because you should" because "I am a programmer and you are not," he is not explaining why something should be done a certain way, he is just being a dick.

Exactly. TUP, I get your point and that really makes sense. I respect that, and it makes sense when you explain it that way. I'm just sick of ident coming in to threads and saying "you suck you're a horrible programmer why would you use `ljk()` instead of `Abc.Edf()` you cunt!?".. This is why people respect you, and not him.

(Jan 13, 2014 12:38 AM)Jazz Wrote: [\[CITE\]](#) Thanks for the source. Seems very handy!
Thank you!

(Jan 13, 2014 07:29 AM)nuk Wrote: [\[CITE\]](#) I know a lot of people look for this code & it's one of the things I'm asked about a lot.... I'm gonna send them to your code :) Great share :)
Thanks man! That's why I like sharing :)

?eumonic - January 13, 2014, 5:35 pm

C# conversion for you...

Figured it'd help. :P

<http://pastebin.com/WzW3zT3J>

Tedd - January 13, 2014, 6:49 pm

(Jan 13, 2014 06:35 PM)Lodestone™ Wrote: [\[CITE\]](#) C# conversion for you...

Figured it'd help. :P

<http://pastebin.com/WzW3zT3J>

Thank you, added to the original post!!