C# Browscap Version

I made a little test with a first C# parser as ConsoleApplication, witch uses Windows native methods, i think C# has no own INI class (if someone knows a better way, please tell us):

For loading the 48048 sections in an array and sort them by length (descending) it takes 796 ms (I think that's ok, because the loading should only happen once), on a search of "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0" the match [Mozilla/5.0 (Windows NT 6.1*WOW64) Gecko/* Firefox/28.0*] was found in 811ms (I think that's bad)
to get the values and copy them to a NameValueCollection it takes 203 ms (that's very bad)

I think the only way to do that with C# is to write an INI-File Class for C# and also write a little function as replacement for the regular expressions.

The following version is only my test version, not all necessary measures for the proper processing were implemented!

ToDo / known issues:

1. Replace the RegExp method with a faster like in my free pascal version
2. Replace the kernel32 implementation of ini-file methods by a native C# class
3. Check for match without comparing numbers

browscap.cs (/code/browscap.cs?mode=download) C-Sharp (5,6 kByte) 24.09.2014 23:13

```
#define DEBUG

using System;
using System.Text;
using System.Runtime.InteropServices;
using System.Collections;
using System.Collections.Specialized;
using System.Text.RegularExpressions;

namespace browscap {
  public class IniFileName {
    [DllImport("kernel32")]
    static extern int GetPrivateProfileSectionNames(
      [MarshalAs(UnmanagedType.LPArray)] byte[] Result, int Size, string FileName);
    [DllImport("kernel32")]
    static extern int GetPrivateProfileSection(
      string Section, [MarshalAs(UnmanagedType.LPArray)] byte[] Result, int Size, string FileName);
    // inserted for prevent matching of group entries https://github.com/browscap/browscap/issues/3
    [DllImport("kernel32")]
    static extern int GetPrivateProfileString(
      string Section, string Key, string Default, StringBuilder Result, int Size, string FileName);

    private static int CompareByLength(string x, string y)
    {
      if (x == null)
      {
        if (y == null)
        {
          return 0;
        }
        else
        {
          return 1;
        }
      }
      else
      {
        if (y == null)
        {
          return -1;
        }
        else
        {
          int retval = x.Length.CompareTo(y.Length);
          if (retval != 0)
          {
            return -retval;
          }
          else
          {
```

```
            return -x.CompareTo(y);
          }
        }
      }
    }

    private string path;
    private string[] sections;

    private string[] GetSectionNames()
    {
      for (int maxsize = 500; true; maxsize *= 2)
      {
        byte[] bytes = new byte[maxsize];
        int size = GetPrivateProfileSectionNames(bytes, maxsize, path);
        if (size < maxsize - 2)
        {
          string Selected = Encoding.ASCII.GetString(bytes, 0, size - (size > 0 ? 1 : 0));
          return Selected.Split(new char[] { '\0' });
        }
      }
    }

    public IniFileName(string INIPath) {
#if DEBUG
      int firstTime = Environment.TickCount;
#endif
      path = INIPath;
      sections = GetSectionNames();
      Array.Sort(sections, CompareByLength);
#if DEBUG
      int lastTime = Environment.TickCount;
      Console.WriteLine("load {0} sections in {1} ms", sections.Length, lastTime - firstTime);
#endif
    }
    // inserted for prevent matching of group entries https://github.com/browscap/browscap/issues/3
    public string IniReadValue(string Section, string Key) {
      StringBuilder temp = new StringBuilder(255);
      int i = GetPrivateProfileString(Section, Key, "", temp, 255, path);
      return temp.ToString();
    }
    private string findMatch(string Agent)
    {
      if (sections != null)
      {
        foreach (string SecHead in sections)
        {  // following line changed for https://github.com/browscap/browscap/issues/438
          if ((SecHead.IndexOf("*", 0) == -1) && (SecHead.IndexOf("?", 0) == -1) && (SecHead ==
Agent))
          {  // inserted for prevent matching of group entries
https://github.com/browscap/browscap/issues/3
            if (IniReadValue(SecHead, "parent") != "DefaultProperties")
            {
              return SecHead;
            }
          }
        }
        foreach (string SecHead in sections)
        {
          try
          {  // following line changed for https://github.com/browscap/browscap/issues/438
            if ((SecHead.IndexOf("*", 0) > -1) || (SecHead.IndexOf("?", 0) > -1))
            {
              if (Regex.IsMatch(Agent, "^" + Regex.Escape(SecHead).Replace(@"\*",
".*").Replace(@"\?", ".") + "$"))
              {
#if DEBUG       // following line shows the matching section and RegEx
                Console.WriteLine(SecHead + ": ^" + Regex.Escape(SecHead).Replace(@"\*",
".*").Replace(@"\?", ".") + "$");
#endif
                return SecHead;
              }
            }
          }
          catch (Exception ex)
```

```
          {
            Console.WriteLine(ex);
          }
        }
        return "*";
      }
      return "";
    }

    public NameValueCollection getValues(string Agent)
    {
#if DEBUG
      Console.WriteLine("search : [" + Agent + "]");
      int firstTime = Environment.TickCount;
#endif
      string match = findMatch(Agent);
#if DEBUG
      int lastTime = Environment.TickCount;
      Console.WriteLine("find match: [{0}] in {1} ms", match, lastTime - firstTime);
      firstTime = Environment.TickCount;
#endif
      NameValueCollection col = new NameValueCollection();
      do
      {
        string[] entries = new string[0];
        bool goon = true;
        for (int maxsize = 500; goon; maxsize *= 2)
        {
          byte[] bytes = new byte[maxsize];
          int size = GetPrivateProfileSection(match, bytes, maxsize, path);
          if (size < maxsize - 2)
          {
            string section = Encoding.ASCII.GetString(bytes, 0, size - (size > 0 ? 1 : 0));
            entries = section.Split(new char[] { '\0' });
            goon = false;
          }
        }
        match = "";
        if (entries.Length > 0)
        {
          foreach (string entry in entries)
          {
            string[] ent = entry.Split(new char[] { '=' });
            if (ent[0] == "Parent")
            {
              match = ent[1];
            }
            else if (col[ent[0]] == null)
            {
              col.Add(ent[0], ent[1]);
            }
          }
        }
      } while (match != "");
#if DEBUG
      lastTime = Environment.TickCount;
      Console.WriteLine("get values: {0} ms", lastTime - firstTime);
#endif
      return col;
    }
  }
}
```

Anwendungbeispiel:

Program.cs (/code/Program.cs?mode=download) C-Sharp (660 bytes) 03.05.2014 16:10

```
using System;
using System.Collections.Specialized;
using browscap;

namespace MainApp {
  class Class1 {
    // The main Console routine.
    [STAThread]
```

```
    static void Main(string[] args) {
      IniFileName bc = new browscap.IniFileName("C:\\browscap\\browscap.ini");
      try {
        NameValueCollection Entry = bc.getValues("Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0)
Gecko/20100101 Firefox/28.0");
        foreach (String s in Entry.AllKeys) {
          Console.WriteLine("{0}={1}", s, Entry[s]);
        }
      } catch (Exception ex) {
        Console.WriteLine("Error:  " + ex);
      }
      Console.ReadLine();
    }
  }
}
```

Output:

```
load 57907 sections in 796 ms
search : [Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0]
Mozilla/5.0 (*Windows NT 6.1*WOW64*) Gecko* Firefox/28.0*:
^Mozilla/5\.0\ \(.*Windows\ NT\ 6\.1.*WOW64.*\)\ Gecko.*\ Firefox/28\.0.*$
find match: [Mozilla/5.0 (*Windows NT 6.1*WOW64*) Gecko* Firefox/28.0*] in 811 ms
get values: 203 ms
Browser_Bits=32
Platform=Win7
Platform_Version=6.1
Platform_Description=Windows 7
Platform_Bits=64
Platform_Maker=Microsoft Corporation
Win32=false
Win64=true
Device_Name=Windows Desktop
Device_Code_Name=Windows Desktop
Comment=Firefox 28.0
Browser=Firefox
Browser_Type=Browser
Browser_Maker=Mozilla Foundation
Version=28.0
MajorVer=28
Frames=true
IFrames=true
Tables=true
Cookies=true
JavaScript=true
JavaApplets=true
CssVersion=3
Device_Maker=Various
Device_Type=Desktop
Device_Pointing_Method=mouse
RenderingEngine_Name=Gecko
RenderingEngine_Version=28.0
RenderingEngine_Description=For Firefox, Camino, K-Meleon, SeaMonkey, Netscape, and other Gecko-
based browsers.
RenderingEngine_Maker=Mozilla Foundation
Browser_Modus=unknown
MinorVer=0
Alpha=false
Beta=false
Win16=false
BackgroundSounds=false
VBScript=false
ActiveXControls=false
isMobileDevice=false
isTablet=false
isSyndicationReader=false
Crawler=false
AolVersion=0
Device_Brand_Name=unknown
```

Autor: Udo Schmal (mailto:udo.schmal@t-online.de), veröffentlicht: 24.05.2014, letzte Änderung: 04.06.2021

© Copyright 2022 Udo Schmal (/)