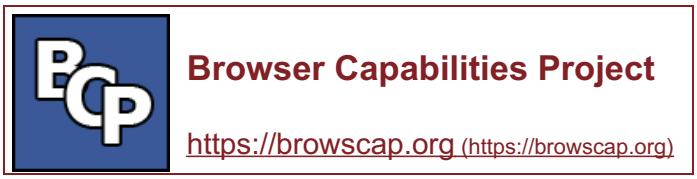
browscap.ini Zugriff über Pascal

Identifizieren des Browsers mit Samt seiner Eigenschaften ist eine tolle Sache, eine lange Zeit lief in Sachen "browscap.ini" nicht mehr viel, doch nun ist das Projekt unter browscap.org (https://browscap.org) zu neuem Leben erwacht und ist wieder "up to date".
Um fehlende User Agents zu melden steht bei GitHub eine Seite zur Verfügung https://github.com/browscap/browscap/issues (https://github.com/browscap/browscap/issues)
Das Team um Thomas Müller und James Titcumb liefern da richtig gute Arbeit!



Identify the Browser an its properties is a good thing, for a long time nothing happens in terms of "browscap.ini", but now the project was awakened to new life. To report any missing user agents join GitHub https://github.com/browscap/browscap/issues (https://github.com/browscap/browscap/issues) .
The team led by Thomas Müller and James Titcumb are making a really good job!

Es gibt zahlreiche Beispiele im Netz um den User Agent des Client-Programms zu ermitteln, es existieren teilweise riesige Tabellen mit Bestandteilen an denen festzustellen sein soll welcher Client es wohl ist der da gerade zugreift. Da allerdings die Anzahl der Clients ständig wächst bedürfen diese Listen ein Ständigen Kontrolle. Einen bewährtes Vorgehen ist das Analysieren des User Agent mit Hilfe einer Methode die bereits seit sehr langer Zeit existiert, die Anwendung der browscap.ini Datei!

There are a lot of programs in all the diferent networks to detect the properties of the client program, mostly there are some huge tables with constituents that should be able to identify the client which is probably the currently accessing there. However, since the number of clients is constantly growing these lists needs a permanent control. A proven approach is to analyze the user agent string by using a method that has been used for a very long time, the browscap.ini file!

Für zahlreiche Systeme existieren bereits Möglichkeiten diese INI zu nutzen, nichts desto Trotz bedarf es aber auch manchmal einer speziellen Lösung. PHP stellt eine native Funktion get_browser() zur Verfügung, die allerdings nicht mehr genutzt werden sollte, stattdessen sollte ein ein Ersatz Browscap-PHP (https://github.com/browscap/browscap-php) heruntergeladen werden. Für ASP gibt es die browscap.dll (MSWC.BrowserType component) und meines Wissens auch die eine oder andere kommerzielle Lösung, auf die ich an dieser Stelle allerdings nicht eingehen geschweige denn das ich sie empfehlen möchte!

For many systems there already exists a ways to use this INI, but sometimes a special solution is needed. PHP provides a native function get_browser() which should no longer be used but instead a replacement Browscap-PHP (https://github.com/browscap/browscap-php) can be downloaded. For ASP, there exists a Browscap.dll (MSWC.BrowserType component) and as I know some other commercial solutions, but at this point I doesn't want to go into it or recommend it!

Der Nachteil der Entwicklung einer Komponente in einer Script-Sprache ist zu meist die Geschwindigkeit und unter Umständen die Ressourcenhungrigkeit, wenn also eine häufig angewandte Methode kompiliert zur Verfügung steht und wenn möglich die Ressourcen an die Applikation und nicht an einen Prozess gebunden sind kann das zu einer gravierenden Zeitersparnis führen. Da aber genau dieses bei stark

frequentierten Websites der Fall ist, habe ich diesen Vorgang in den ISAPI-Filter verlegt und kopiere die ermittelten Daten direkt in das Session-Objekt, wo sie dann für die komplette Session zur Verfügung stehen.

The disadvantage of the development of a component in a scripting language is usually the performance loss and possibly the used memory size, so if a frequently used method is available in a compiled version and further if the resources are attached to the application and not to a process, then that can be a serious time saver. Exactly this is the case why I have moved this process in an ISAPI filter and copy the data obtained directly in the Session object, where they are available for the entire session, especially for Websites with much traffic.

browscap.pas Pascal (6,64 kByte) 10.08.2014 14:13

```
//
//******************************************************************************
//  Title.............. :  browscap Components Library
//
//  Modulname ......... :  browscap.pas (browscap.ini Component)
//  Type .............. :  Unit (Component Library)
//  Author ............ :  Udo Schmal
//  Development Status  :  10.08.2014
//  Operating System .. :  Win32/64
//  IDE ............... :  Lazarus
//******************************************************************************

unit browscap;
{$ifdef fpc}
  {$mode objfpc}
{$endif}
{$H+}
interface

uses Classes, SysUtils, IniFiles;

type
  PStringList = ^TStringList;
  TBrowscap = class(TIniFile)
  private
    FPValueList: PStringList;
    FSections: TStringList;
    function matchWithWildcard(ASection: string; const AAgent: string;
ignoreNum: boolean = false): boolean;
    function determineSection(const AAgent: string): string;
    procedure readSection(const ASection: string);
    function GetVersion(): integer;
    function GetRelease(): string;
  public
    constructor Create(const AIniFile: string); // path to browscap.ini file
    destructor Destroy(); override;
    procedure GetUserAgentInfo(const AUserAgent: string; var AValueList:
TStringList);
    property Version: integer read GetVersion; // browscap.ini version
    property Release: string read GetRelease; // browscap.ini release
```

```
    property Sections: TStringList read FSections;
  end;

implementation

constructor TBrowscap.Create(const AIniFile: string);
  procedure QuickSortSections(iLo, iHi: integer);
  var Lo, Hi, Mid: integer;
  begin
    Lo := iLo;
    Hi := iHi;
    Mid := length(FSections.Strings[(Lo + Hi) div 2]);
    repeat
      while length(FSections.Strings[Lo]) > Mid do Inc(Lo);
      while length(FSections.Strings[Hi]) < Mid do Dec(Hi);
      if Lo <= Hi then
      begin
        FSections.Exchange(Lo, Hi);
        Inc(Lo);
        Dec(Hi);
      end;
    until Lo > Hi;
    if Hi > iLo then QuickSortSections(iLo, Hi);
    if Lo < iHi then QuickSortSections(Lo, iHi);
  end;
//var i: integer;
begin
  inherited Create(AIniFile);
  FPValueList := nil;
  FSections := TStringList.Create;
  FSections.Sorted := false;
  ReadSections(FSections);
  FSections.BeginUpdate;
  try
    FSections.Delete(0); // no 'GJK_Browscap_Version'
    FSections.Delete(0); // no 'DefaultProperties'
    FSections.Delete(FSections.Count-1); // no '*'
// the following while statement can be commented if the fix is done in
determineSection
// if uncomment the construction takes mutch more time but each check in
determineSection is mutch faster
//    i := 0;
//    while i < FSections.Count-1 do
//    begin
//      if (pos('*',FSections.Strings[i])=0) and
(pos('?',FSections.Strings[i])=0) then
//        if ReadString(FSections.Strings[i], 'parent', '') <>
'DefaultProperties' then // fix for group entry
//          FSections.Delete(i)
//        else
//          inc(i)
```

```
//      else
//        inc(i);
//    end;
    QuickSortSections(0, FSections.Count-1);
  finally
    FSections.EndUpdate;
  end;
end;

destructor TBrowscap.Destroy;
begin
  FPValueList := nil;
  FSections.Free;
  inherited Destroy;
end;

function TBrowscap.matchWithWildcard(ASection: string; const AAgent: string;
ignoreNum: boolean = false): boolean;
var
  pAgent: array [0..255] of Char;
  pSection: array [0..255] of Char;
  function MatchSection(agent, section: PChar): boolean;
  begin
    if StrComp(section, PChar('*')) = 0 then
      result := true
    else if (agent^ = Chr(0)) or (section^ = Chr(0)) then
      result := (agent^ = Chr(0)) and (section^ = Chr(0))
    else
      case section^ of
        '*': if MatchSection(agent, @section[1]) then
               result := true
             else
               result := MatchSection(@agent[1], section);
        '?': result := MatchSection(@agent[1], @section[1]);
        else
          if agent^ = section^ then
            result := MatchSection(@agent[1], @section[1])
          else if ignoreNum and (agent^ in ['0'..'9']) and (section^ in
['0'..'9']) then
            result := MatchSection(@agent[1], @section[1])
          else
            result := false;
      end;
  end;
begin
  result := false;
  StrPCopy(pSection, ASection);
  StrPCopy(pAgent, AAgent);
  result := MatchSection(pAgent, pSection);
end;
```

```
function TBrowscap.determineSection(const AAgent: string): string;
var
  i: integer;
  s: string;
begin
  result := '*';
  // UserAgent try for exact match
  if (FSections.indexOf(AAgent) <> -1) then
// the following if statement can be commented if the fix is allready done in
the costructor
    if ReadString(AAgent, 'parent', '') <> 'DefaultProperties' then // fix for
group entry
      result := AAgent;
  // UserAgent search using wildcard sections
  if result = '*' then // not found jet
  begin
    for i := 0 to (FSections.count - 1) do
    begin
      s := FSections[i];
      if (length(AAgent)>length(s)-2) and ((pos('*',s)>0) or (pos('?',s)>0))
then
        if matchWithWildcard(s, AAgent) then
        begin
          result := s;
          break;
        end;
    end;
    // UserAgent partial match with wildcards and ignore numbers
    if result = '*' then // not found jet
      for i := 0 to (FSections.count - 1) do
      begin
        s := FSections[i];
        if (length(AAgent)>length(s)-2) then
          if matchWithWildcard(s, AAgent, true) then
          begin
            result := s;
            break;
          end;
      end;
  end;
end;

procedure TBrowscap.readSection(const ASection: string);
var
  parentSection: string;
  sectionValues: TStringList;
  i: integer;
begin
  parentSection := ReadString(ASection, 'parent', '');
  if ((parentSection <> '') and (parentSection <> ASection)) then
    readSection(parentSection);
```

```
  sectionValues := TStringList.create;
  try
    readSectionValues(ASection, sectionValues);
    for i:=0 to sectionValues.Count-1 do
      if sectionValues.Strings[i][1]<>';' then
        FPValueList^.Values[sectionValues.names[i]] :=
sectionValues.ValueFromIndex[i];
  finally
    sectionValues.Free;
  end;
end;

function TBrowscap.GetVersion(): integer;
begin
  result := ReadInteger('GJK_Browscap_Version', 'Version', 0);
end;

function TBrowscap.GetRelease(): string;
begin
  result := ReadString('GJK_Browscap_Version', 'Released', '');
end;

procedure TBrowscap.GetUserAgentInfo(const AUserAgent: string; var AValueList:
TStringList);
var sSection: string;
begin
  if Assigned(AValueList) then
  begin
    FPValueList := @AValueList;
    sSection := determineSection(AUserAgent);
    readSection(sSection);
  end;
end;

end.
```

Nun als Beispiel eine fertige Routine zum Testen der Zeit, die CriticalSection ist zwar nicht erforderlich, aber da sie in der Servererweiterung auch erforderlich ist, habe ich sie direkt erhalten.

Now an Example of the usage of the Class to check the performance, the CriticalSection is not used for this Example but in the server extension it is used.

browscap.lpr Pascal (1,74 kByte) 29.04.2014 23:18

```
program browscap;

uses SysUtils, Classes, browscap;

var
// make it thread save
  BrowscapIniLock: TRTLCriticalSection;
// the object
  BrowscapIni: TBrowscap;
```

```
// get UserAgent Values
  ValueList: TStringList;
  sUserAgentString: string;
  i: integer;
// get used time
  StartTime: QWord;
begin
// onStart
  StartTime := GetTickCount64();
{$ifdef fpc}
  InitCriticalSection(BrowscapIniLock); //SyncObjs
{$else}
  InitializeCriticalSection(BrowscapIniLock); //Windows
{$endif}
  BrowscapIni := TBrowscap.Create(ExtractFilePath(ParamStr(0)) +
'browscap.ini');
  WriteLn('time read browscap.ini: ' + IntToStr(GetTickCount64() - StartTime)
+ ' ms');

  StartTime := GetTickCount64();
  sUserAgentString := 'Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0)
Gecko/20100101 Firefox/28.0';
//  sUserAgentString := 'DoCoMo/2.0 N905i(c100;TB;W24H16) (compatible;
Googlebot-Mobile/2.1; +http://www.google.com/bot.html)';
//  sUserAgentString := 'Mozilla/5.0 (compatible; uMBot-LN/1.0; mailto:
crawling@ubermetrics-technologies.com)';
  ValueList := TStringList.Create;
  EnterCriticalSection(BrowscapIniLock);
  try
    BrowscapIni.GetUserAgentInfo(sUserAgentString, ValueList);
  finally
    LeaveCriticalSection(BrowscapIniLock);
  end;
  WriteLn('time to getValues: ' + IntToStr(GetTickCount64() - StartTime) + '
ms');

  // list Browsecap.ini UserAgent Values
  for i:=0 to ValueList.Count-1 do
    WriteLn(ValueList.Strings[i]);
  ValueList.Free;

//onEnd
  StartTime := GetTickCount64();
  BrowscapIni.Free;
{$ifdef fpc}
  DoneCriticalsection(BrowscapIniLock); //SyncObjs
{$else}
  DeleteCriticalSection(BrowscapIniLock); //Windows
{$endif}
  WriteLn('time to close browscap.ini: ' + IntToStr(GetTickCount64() -
StartTime) + ' ms');
```

```
end.
```

Und nun die Ausgabe, an ihr ist ganz einfach zu erkennen, das es nur Sinn macht die Full Version der browscap.ini zu nutzen wenn der eigentliche Index erhalten bleiben kann, das Laden der INI-Datei in das INI-Object und das Sortieren der Section-Liste schlägt mit satten 1123 ms zu Buche. Natürlich ist seit Kurzen auch das finden des passenden Eintrags schwerer geworden da gerade in letzter Zeit die Anzahl der Sektionen auf weit über 50.000 gestiegen ist. 32 ms hört sich nicht so schlimm an, aber auf stark frequentierten Servern kann das zu erheblichen Problemen führen.

In den letzten Optimierungen wurden Regular Expressions entfernt, eine Quicksort Routine eingesetzt, an Stelle der TStringList eigenen CustomSort Variante, das ist nicht so schön aber insgesamt ergab es eine enorme Zeitersparnis!

```
time read browscap.ini: 1123 ms
time to getValues: 32 ms
Comment=Firefox 28.0
Browser=Firefox
Browser_Type=Browser
Browser_Bits=32
Browser_Maker=Mozilla Foundation
Browser_Modus=
Version=28.0
MajorVer=28
MinorVer=0
Platform=Win7
Platform_Version=6.1
Platform_Description=Windows 7
Platform_Bits=64
Platform_Maker=Microsoft Corporation
Alpha=false
Beta=false
Win16=false
Win32=false
Win64=true
Frames=true
IFrames=true
Tables=true
Cookies=true
BackgroundSounds=false
JavaScript=true
VBScript=false
JavaApplets=true
ActiveXControls=false
isMobileDevice=false
isTablet=false
isSyndicationReader=false
Crawler=false
CssVersion=3
AolVersion=0
Device_Name=Windows Desktop
Device_Maker=Various
Device_Type=Desktop
```

```
Device_Pointing_Method=mouse
Device_Code_Name=Windows Desktop
Device_Brand_Name=unknown
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
Parent=Firefox 28.0
time to close browscap.ini: 141 ms
```

Das nutzen dieser Routine in einem ActiveX Objekt, in einer ASP-Komponente wie der browscap.dll oder auch einem CGI macht daher nicht wirklich Sinn. Das Erstellen eines ISAPI-Filters oder eines Apache-Modules hingegen würde sich anbieten da in diesen Fällen die vorbereiteten Daten immer wieder genutzt werden können. Des weiteren sollten die ermittelten Werte dann in Session-Variablen gespeichert werden, damit sie nicht immer wieder bei erneuten zugriffen ermittelt werden müssen.

In meinem Auftritt also hier nutze ich dann auch noch eine optimierte Liste und nicht die Standard TStringList von Object-Pascal, leider wird der Geschwindigkeitsgewinn durch ausgiebige Überprüfungen und Logging aufgefressen, aber nur dadurch ist das Aufspüren von Fehlern möglich, siehe browscap.ini issues.

Autor: Udo Schmal, veröffentlicht: 25.04.2014, letzte Änderung: 29.08.2024