

User Agent String Analyse

Web-Clients übertragen im HTTP-Header einen User-Agent String der im Regelfall Informationen zum Client an den Server übergibt. Schaut man grob auf diese Zeichenketten könnte man glauben diese Zeichenketten unterliegen keiner Normung, doch das stimmt nicht, die meisten Varianten lassen sich logisch auflösen und genau das ist erforderlich für eine ordentliche Analysesoftware. Wenn man im Netz nach Varianten der User-Agent String Analysen sucht findet man immer wieder die Anwendung von Regular-Expressions, doch diese Variante ist bei einer solchen Flut von User-Agents aus meiner Sicht nicht mehr Sinnvoll.

An dieser Stelle möchte ich kurz mein Konzept erläutern.

Zur Analyse eines User-Agent Strings benötigt man eigentlich nur wenige Regeln, immer wiederkehrende Elemente sind z.B. Name/Version in der Regel durch einen Slash getrennt oder auch Kommentare immer in runden Klammern. Zerlegt man diese Zeichenketten erhält man meist kurze Bezeichnungen (Name) wie z.B. Firefox, Trident, AppleWebKit, ... leider gibt es natürlich auch Ausnahmen aber selbst diese lassen sich herauslösen. Als ersten sollte man alle Regeln kennen, im groben kann man sagen außerhalb der Klammern wird ein Teilstring "Name-Version)" nur durch ein Leerzeichen nach einem Versions-Wert oder durch den Start eines Kommentars die öffnende runde Klammer "(" beendet, innerhalb der Klammern werden die Teilstrings durch Semikolon und auch durch Komma getrennt.

Die daraus resultierenden Teilstrings geben die unterschiedlichsten Informationen zurück, legt man hierzu eine Liste an lässt sich das Geheimnis eines User-Agent Strings schnell entschlüsseln, zwar wird diese Liste auch einige Tausend Zeilen lang jedoch wird keine riesige Anzahl von Regular Expressions benötigt.

Umsetzung

1. Schritt: Liste von User-Agent Strings als Testdatenmenge erzeugen

Als erstes benötigen wir eine möglich lange Liste von User-Agent Strings, am Besten solche die wir später auch analysieren möchten, damit schon mal diese alle zu einem Ergebnis führen. Im Internet stehen zahlreiche Listen online ich habe mich dazu entschlossen, einmal kurz alle User-Agent Strings der letzten sieben Jahre aus meinen Logfiles zu extrahieren, natürlich ohne Doublets.

[collectUAs.lpr](#) Pascal (3,42 kByte) 03.06.2017 14:52

```
program collectUAs;
{$ifdef FPC}
 {$mode objfpc}
{$endif}
{$H+}
uses SysUtils, Classes;

var
 sr: TSearchRec;
 i, f, o, ua: integer;
 sPath, line, s: string;
 fields, useragents: TStringList;
begin
 if (ParamStr(1)='') or not DirectoryExists(ParamStr(1)) then
```

```

begin
  WriteLn('Collect UserAgent Strings');
  Writeln('Usage: collectUAs [pathToTheLogFileFolder]');
end
else
begin
  sPath := ParamStr(1);
  fields := TStringlist.Create();
try
  fields.Delimiter := ' ';
  fields.QuoteChar := #$0;
// create the storage
  useragents := TStringlist.Create();
try
  if FindFirst(sPath + '*.log', faAnyFile, sr) = 0 then
    repeat
      // loop through the logfiles
      with TStringList.Create() do
        try
          LoadFromFile(sPath + sr.Name);
          for i:=0 to Count-1 do
            begin
              // loop through the lines of the logfile
              line := Strings[i];
              if (trim(line)<>'') then // jump over empty lines
              begin
                if (pos('#Fields:', line)=1) then
                  // read the field list
                  with fields do
                    try
                      DelimitedText := line;
                      for f:=0 to Count-1 do
                        // for this case we need only the user-agent
                        if Strings[f]='cs(User-Agent)' then
                          begin
                            ua := f-1;
                            break;
                          end;
                        f := Count-1;
                      finally
                        clear;
                      end
                    else if not (pos('#',line)=1) then // ignore all other
comments
                      // get only the Values
                      with fields do
                        try
                          // replace possible errors (an empty field)
                          DelimitedText := StringReplace(line, ' ', ' - ', [rfReplaceAll]);

```

```

        // only check the lines that have the right number of
fields

        if (Count = f) then
begin
    // get the User-Agent Value
    s := Strings[ua];
    // IIS replaces a space by a plus, replace "+" by " "
    s := StringReplace(s, '++', ' '#9, [rfReplaceAll]);
    s := StringReplace(s, '+(+http', '('#9'http',
[rfReplaceAll]);

    s := StringReplace(s, '+', ' ', [rfReplaceAll]);
    s := StringReplace(s, #9, '+', [rfReplaceAll]);
    // check if user-agent is already in the storage
    o := useragents.IndexOf(s);
    if o = -1 then
begin
    // if not, add the new user-agent
    writeln(s);
    useragents.Add(s);
end;
end;
finally
    clear;
end;
end;
end;
end;
finally
    Free;
end;
// save useragent.txt in local path
useragents.SaveToFile(ExtractFilePath(ParamStr(0)) +
'useragents.txt');

until FindNext(sr) <> 0;
FindClose(sr);
finally
    useragents.Free;
end;
finally
    fields.Free;
end;
end;
end.

```

Aufruf: collectUAs [pathToLogFileFolder] bei mir C:\inetpub\logfiles\ bitte am Ende den Backslash "\\" nicht vergessen, nach dem erfolgreichen start sollten dann in der Console die User-Agents die gerade gefunden werden auch ausgegeben werden, am Ende liegt im Programm-Ordner eine Datei useragents.txt sie enthält dann alle gefundenen Einträge.

[collectUAs.zip kompilierte Win32 Exe-Datei](#) (71,69 kByte) 30.12.2018 21:29

2. Schritt: Liste in Bestandteile (Name/Version) zerlegen

Nun da wir bereits eine gewisse Testmenge zur Verfügung haben (ich habe noch viele weitere aus dem Internet gefundene hinzugefügt) können wir die wichtigen Teile ermitteln, dabei wollen wir die Versionsnummern und die Sprachen ignorieren. Für die Sprachen werden wir später eine Sprachen und Länderliste anlegen, die Versionsnummern sollen variabel bleiben da wir nicht jede neue Browser-Version in unsere Liste aufnehmen wollen sondern den Wert direkt aus dem User-Agent String nehmen.

[collectUAParts.lpr](#) Pascal (9,29 kByte) 03.06.2017 23:00

```
program collectUAParts;
{$ifdef FPC}
 {$mode objfpc}
{$endif}
{$H+}
uses SysUtils, Classes;

type TStates = (STATE_UNDEFINED, STATE_NAME, STATE_VERSION, STATE_COMMENT,
                STATE_COMMENT_VERSION, STATE_LANGUAGE);

var
  sName, sVersion, sLang: string;
  parts: TStringList;

procedure checkPart();
const
  lang: array[0..186] of string = (
    'aa', 'ab', 'ae', 'af', 'ak', 'am', 'an', 'ar', 'as', 'av', 'ay', 'az',
    'ba', 'be', 'bg', 'bh', 'bi', 'bm', 'bn', 'bo', 'br', 'bs', 'ca', 'ce',
    'ch', 'co', 'cr', 'cs', 'cu', 'cv', 'cy', 'da', 'de', 'dv', 'dz', 'ee',
    'el', 'en', 'eo', 'es', 'et', 'eu', 'fa', 'ff', 'fi', 'fj', 'fo', 'fr',
    'fy', 'ga', 'gd', 'gl', 'gn', 'gu', 'gv', 'ha', 'he', 'hi', 'ho', 'hr',
    'ht', 'hu', 'hy', 'hz', 'ia', 'id', 'ie', 'ig', 'ii', 'ik', 'io', 'is',
    'it', 'iu', 'ja', 'jv', 'ka', 'kg', 'ki', 'kj', 'kk', 'kl', 'km', 'kn',
    'ko', 'kr', 'ks', 'ku', 'kv', 'kw', 'ky', 'la', 'lb', 'lg', 'li', 'ln',
    'lo', 'lt', 'lu', 'lv', 'mg', 'mh', 'mi', 'mk', 'ml', 'mn', 'mo', 'mr',
    'ms', 'mt', 'my', 'na', 'nb', 'nd', 'ne', 'ng', 'nl', 'nn', 'no', 'nr',
    'nv', 'ny', 'oc', 'oj', 'om', 'or', 'os', 'pa', 'pi', 'pl', 'ps', 'pt',
    'qu', 'rc', 'rm', 'rn', 'ro', 'ru', 'rw', 'sa', 'sc', 'sd', 'se', 'sg',
    'sh', 'si', 'sk', 'sl', 'sm', 'sn', 'so', 'sq', 'sr', 'ss', 'st', 'su',
    'sv', 'sw', 'ta', 'te', 'tg', 'th', 'ti', 'tk', 'tl', 'tn', 'to', 'tr',
    'ts', 'tt', 'tw', 'ty', 'ug', 'uk', 'ur', 'uz', 've', 'vi', 'vo', 'wa',
    'wo', 'xh', 'yi', 'yo', 'za', 'zh', 'zu');
  country: array[0..241] of string = (
    'AF', 'AL', 'DZ', 'AS', 'AD', 'AO', 'AI', 'AQ', 'AG', 'AR', 'AM', 'AW',
    'AU', 'AT', 'AZ', 'BS', 'BH', 'BD', 'BB', 'BY', 'BE', 'BZ', 'BJ', 'BM',
    'BT', 'BO', 'BA', 'BW', 'BV', 'BR', 'IO', 'VG', 'BN', 'BG', 'BF', 'BI',
    'KH', 'CM', 'CA', 'KV', 'KY', 'CF', 'TD', 'CL', 'CX', 'CC', 'CO', 'KM',
    'CG', 'CD', 'CK', 'CR', 'CI', 'HR', 'CU', 'CY', 'CZ', 'DK', 'DJ', 'DM',
    'DO', 'TP', 'EC', 'EG', 'SV', 'GQ', 'ER', 'EE', 'ET', 'FK', 'FO', 'FJ',
    'FI', 'FR', 'GF', 'PF', 'TF', 'GA', 'GM', 'GE', 'DE', 'GH', 'GI', 'GB'
```

```

'GR', 'GL', 'GD', 'GP', 'GU', 'GT', 'GN', 'GW', 'GY', 'HT', 'HM', 'VA',
'HN', 'HK', 'HU', 'IS', 'RE', 'IN', 'ID', 'IR', 'IQ', 'IE', 'IL', 'IT',
'JM', 'JP', 'JO', 'KZ', 'KE', 'KI', 'XK', 'KW', 'KG', 'LA', 'LV', 'LB',
'LS', 'LR', 'LY', 'LI', 'LT', 'LU', 'MO', 'MK', 'MG', 'MW', 'MY', 'MV',
'ML', 'MT', 'MH', 'MQ', 'MR', 'MU', 'YT', 'MX', 'FM', 'MD', 'MC', 'MN',
'ME', 'MS', 'MA', 'MZ', 'MM', 'NA', 'NR', 'NP', 'NL', 'AN', 'NC', 'NZ',
'NI', 'NE', 'NG', 'NU', 'NF', 'KP', 'MP', 'NO', 'OM', 'PK', 'PW', 'PS',
'PA', 'PG', 'PY', 'CN', 'PE', 'PH', 'PN', 'PL', 'PT', 'PR', 'QA', 'SM',
'RO', 'RU', 'RW', 'KN', 'LC', 'PM', 'VC', 'WS', 'ST', 'SA', 'SN', 'RS',
'SC', 'SL', 'SG', 'SK', 'SI', 'SB', 'SO', 'ZA', 'GS', 'KR', 'ES', 'LK',
'SH', 'SD', 'SR', 'SJ', 'SZ', 'SE', 'CH', 'SY', 'TW', 'TJ', 'TZ', 'TH',
'TG', 'TK', 'TO', 'TT', 'TN', 'TR', 'TM', 'TC', 'TV', 'UG', 'UA', 'AE',
'UK', 'US', 'UY', 'UM', 'UZ', 'VU', 'VE', 'VN', 'VI', 'WF', 'EH', 'YE',
'ZM', 'ZW');

function isLang(AString: string): boolean;
var i: integer;
begin
  for i := Low(lang) to High(lang) do
    if lang[i] = AString then
      exit(true);
    result := false;
  end;
end;

function isCountry(AString: string): boolean;
var i: integer;
begin
  for i := Low(country) to High(country) do
    if country[i] = AString then
      exit(true);
    result := false;
  end;
end;

function isLangCountry(AString: string): boolean;
begin
  if (length(AString) = 2) or ((length(AString) = 5) and (AString[3] = '-')) then
    begin
      if (length(AString) = 5) and not isCountry(copy(AString, 4, 2)) then
        exit(false);
      result := isLang(copy(AString, 1, 2));
    end
    else
      result := false;
  end;
end;

begin
  sName := trim(sName);
  sVersion := trim(sVersion);
  if sName <> '' then
    begin
      if (parts.IndexOf(sName) = -1) and
        not (pos('http', sName)=1) and
        not isLangCountry(sName) then

```

```

begin
  // if not exists add the new part to list
  writeln(sName);
  // we only collect the names the Version is variable
  parts.Add(sName);
end;
end;
sName := '';
sVersion := '';
sLang := ''; //ignore
end;

var
o, i, len: integer;
sFile, sUserAgent: string;
useragents: TStringList;
state, fallbackState: TStates;
c: char;
begin
  sFile := ExtractFilePath(ParamStr(0)) + 'useragents.txt';
  if FileExists(sFile) then
    begin
      useragents := TStringlist.Create();
      try
        useragents.LoadFromFile(sFile);
        parts := TStringList.Create();
        try
          for o:=0 to useragents.Count-1 do
            begin
              // get a user-agent from the generated list
              sUserAgent := useragents.Strings[o];
              // check if it can be a user-agent
              if not (sUserAgent[1] in ['A'..'Z', 'a'..'z', '[']) then
                continue;
              // start parsing
              len := Length(sUserAgent);
              state := STATE_UNDEFINED;
              fallbackState := STATE_UNDEFINED;
              for i:=1 to len do
                begin
                  c := sUserAgent[i];
                  // the state is undefined after each part and at the beginning
                  if (state = STATE_UNDEFINED) then
                    begin
                      if (c = '(') then
                        // start of comment
                        state := STATE_COMMENT
                      else if (c = '[') then
                        // start of old language part
                        state := STATE_LANGUAGE

```

```

        else if not (c in [' ', '+', ')', ';']) then // ignores this at
the beginnig of a new part
begin
    // start of a name part
    state := STATE_NAME;
    sName := c;
end
end
else if (state = STATE_NAME) then
begin
    if (c = '/') then
        // start of the version part
        state := STATE_VERSION
    else if (c = '+') then
begin
        // something new starts
        checkPart();
        state := STATE_UNDEFINED
end
    else if (c = '(') then
begin
        checkPart();
        // start of comment
        state := STATE_COMMENT;
end
else
    // go on writing the name
    sName := sName + c;
end
else if (state = STATE_VERSION) then
begin
    if (c = ' ') then
begin
        // something new starts
        state := STATE_UNDEFINED;
        checkPart();
end
    else if (c = '/') then
begin
        // the first slash was part of the name (fix)
        sName := sName + '/' + sVersion;
        sVersion := '';
end
    else
        // go on writing the version
        sVersion := sVersion + c;
end
else if (state in [STATE_COMMENT, STATE_COMMENT_VERSION]) then
begin
    if (c = ')') then

```

```

begin
    // the comment ends, something new starts
    checkPart();
    state := fallbackState;
end
else if (c = '(') then
begin
    // there is a comment inner comment
    checkPart();
    fallbackState := STATE_COMMENT;
    state := STATE_COMMENT;
end
else if (c in [';', ',', '']) then
begin
    // next comment
    checkPart();
    state := STATE_COMMENT;
end
else if (c in ['+', ' ']) and (sName = '') then
    // ignore + and space of the beginning of a comment
else if (c = '/') and (pos('http:', sName)=0) and
(pos('https:', sName)=0) then
    // start with version in comment part
    state := STATE_COMMENT_VERSION
else if (c = ':') and (sName = 'rv') then
    // rv is a special case it is separated by a colon
    // start with version in comment part
    state := STATE_COMMENT_VERSION
else
begin
    if (state = STATE_COMMENT_VERSION) then
        // go on writing the comment version
        sVersion := sVersion + c
    else
        // go on writing the comment name
        sName := sName + c;
    end;
end
else if (state = STATE_LANGUAGE) then
begin
    if (c = ']') then
    begin
        checkPart();
        state := STATE_UNDEFINED
    end
    else
        // go on reading the language
        sLang := sLang + c;
    end;
end;

```

```

    // analyse the last part
    checkPart();

    end;
    finally
        parts.Sort;
        parts.SaveToFile(ExtractFilePath(ParamStr(0)) + 'useragents-
parts.txt');
        parts.Free;
    end;
    finally
        useragents.Free;
    end;
end;

```

Einfach im selben Verzeichnis starten, dann wird eine Datei useragents-parts.txt angelegt, während das sammeln der User-Agent Strings zeitaufwändig ist dauert dieser Vorgang nur wenige Sekunden und das Schöne ist die Liste wird in der Regel sogar wesentlich kürzer.

[collectUAParts.zip kompilierte Win32 Exe-Datei](#)
(111,32 kByte) 30.12.2018 21:29

3. Schritt: User-Agent Class oder Record Definieren

An dieser Stelle macht es Sinn sich darüber Gedanken zu machen, welche Informationen wir aus den Bestandteilen bekommen und welche Informationen wir brauchen oder uns wünschen. Mit unserer generierten Liste können wir bereits einen Begriff davon bekommen aus welchen Bereichen unsere Informationen sind. Die drei mehr oder minder großen Bereiche sind Client(Browser/Crawler/...), Hardware und Betriebssystem, aber wie teilen wir die Informationen sinnvoll auf? Eine Erste Liste:

- ClientType (Browser, Bot, Search, Check, Crawler, ...)
- ClientName (Firefox, Internet Explorer, Opera, Safari, Chrome, ...)
- ClientVersion (Versionsnummer des Browsers aus dem User-Agent String)
- ClientRenderingEngineName (Gecko, Trident, AppleWebKit, ...)
- ClientRenderingEngineVersion (Versionsnummer der Rendering Engine aus dem User-Agent String)
- ClientAddons (z.B. Plugins wie: Ask Toolbar, Hotbar, Mail.ru Agent - Instant Messenger / VoIP, ...)
- ClientBasedOn (KHTML, oder auch IE beim Avant Browser, Firefox beim SeaMonkey, ...)
- ClientComponents (Liste der beteiligten Komponenten wenn vorhanden mit Version, MS.NET SDK, Java Standard Library, libcurl, zlib, libssh, ...)
- ClientInfo (zusätzliche Informationen, z.B. Hyperlink bei Suchmaschinen, ...)
- ClientLanguage (Interface Sprache wie z.B. de-DE, en-GB, nl-NL, ...)
- PlatformType (Smartphone, Tablet, PC, ...)
- PlatformName (iPhone, iPad, HTC Desire, HTC One, SAMSUNG SM-J500FN, Dell, ...)
- PlatformArchitecture (x86, ARM, ARM/64, MIPS, ...)
- OperatingSystemName (Windows, Linux, OS-X, Android, ...)
- OperatingSystemDistribution (Debian, Ubuntu, ...)
- OperatingSystemVersion (Versionsnummer des Betriebssystems aus dem User-Agent String, für Windows Aufschlüsselung z.B 2000, XP, 7, Vista, 10, ...)

Um das Record zu füllen benötigen wir eine Sammlung der Begriffe und deren Bedeutung [ua.xml](#) und natürlich muss noch die Routine die wir Schritt 2 benutzt haben darauf angepasst werden um die Begriffe zu vergleichen und anschließend an der richtigen Stelle im Record einzusetzen. Ich habe bewusst keine Tabelle im Programm Code direkt gefüllt da ich anschließend die XML-Daten auch noch nutzen kann wenn ich die Routine in anderen Sprachen schreibe. Sicherlich fehlen in dieser Liste auch noch sehr viele Begriffe, aber die werden in Kürze nachgepflegt, für einen Test sollte es allerdings reichen. (4,12 kByte) 30.12.2018 21:29

[ua.xml](#)

XML (4,12 kByte) 15.06.2017 16:00

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<root>
  <item token="Mozilla" type="ignore" />
  <item token="compatible" type="ignore" />
  <item token="U" type="ignore" />
  <item token="MSIE /" name="Internet Explorer" info="Microsoft"
type="browser" />
  <item token="Trtoken" name="Trtoken" type="engine" />
  <item token="Firefox" name="Firefox" info="Mozilla, ex Firebird"
type="browser" />
  <item token="Gecko" name="Gecko" type="engine" />
  <item token="Chrome" name="Chrome" info="Google Chrome" type="browser" />
  <item token="Safari" name="Safari" info="Apple" type="browser" />
  <item token="Mobile Safari" name="Mobile Safari" type="browser" />
  <item token="Mobile" name="Mobile" type="browser" />
```

```
<item token="AppleWebKit" name="AppleWebKit" type="engine" />
<item token="Opera*" name="Opera" type="browser" />
<item token="Presto" name="Presto" type="engine" />
<item token="SeaMonkey" name="SeaMonkey" type="browser" />
<item token="Windows 4.90" name="Windows ME" type="os" />
<item token="Windows 95" name="Windows 95" type="os" />
<item token="Windows 98" name="Windows 98" type="os" />
<item token="Windows ARM" name="Windows RT" type="os" />
<item token="Windows CE" name="Windows CE" type="os" />
<item token="Windows NT 10.0" name="Windows 10" type="os" />
<item token="Windows NT 5.0" name="Windows 2000" type="os" />
<item token="Windows NT 5.1" name="Windows XP" type="os" />
<item token="Windows NT 5.2" name="Windows XP SP2" type="os" />
<item token="Windows NT 6.0" name="Windows Vista" type="os" />
<item token="Windows NT 6.1" name="Windows 7" type="os" />
<item token="Windows NT 6.2" name="Windows 8" type="os" />
<item token="Windows NT 6.3" name="Windows 8.1" type="os" />
<item token="Windows NT 6.4" name="Windows 10" type="os" />
<item token="Windows NT3.51" name="Windows NT 3.11" type="os" />
<item token="Windows NT4.0" name="Windows NT 4.0" type="os" />
<item token="Windows Phone" name="Windows Phone" type="os" />
<item token="Windows" type="ignore" />
<item token="SUSE" name="SUSE" type="distribution" />
<item token="Debian-/" name="Debian" type="distribution" />
<item token="RedHat" name="RedHat" type="distribution" />
<item token="Ubuntu" name="Ubuntu" type="distribution" />
<item token="Edition Linux Mint" name="Mint" type="distribution" />
<item token="Linux Mint" name="Mint" type="distribution" />
<item token="Mint" name="Mint" type="distribution" />
<item token="Linux /" name="Linux" type="os" />
<item token="Linux" name="Linux" type="os" />
<item token="Mac OS" name="Mac OS" type="os" />
<item token="iOS" name="iOS" type="os" />
<item token="CPU OS /" name="iOS" type="os" />
<item token="CPU iPhone OS /" name="iOS" type="os" />
<item token="Android Browser" name="Android Browser" type="browser" />
<item token="Android /" name="Android" type="os" />
<item token="AMD64" name="AMD64" type="cpu" />
<item token="ARM" name="ARM" type="cpu" />
<item token=".NET CLR /" name="MS.NET SDK" type="component" />
<item token=".NET4.//" name=".NET4" type="component" />
<item token="iPad" name="iPad" type="device" />
<item token="iPhone" name="iPhone" type="device" />
<item token="iPod" name="iPod" type="device" />
<item token="Google Page Speed Insights" name="Google Page Speed Insights" type="check" />
<item token="Google Page Speed" name="Google Page Speed" type="check" />
<item token="Google Talk" name="Google instant messenger" type="browser" />
<item token="Google WAP Proxy" name="Google WAP Proxy" type="proxy" />
```

```

<item token="Google Web Preview" name="Google Web Preview" type="check" />
<item token="Google-SearchByImage" name="Google SearchByImage" type="bot"
/>
<item token="Google-Site-Verification" name="Google Site Verification"
type="check" />
<item token="Googlebot" name="Google robot" type="bot" />
<item token="Googlebot-Image" name="Google image crawler" type="bot" />
<item token="Googlebot-Mobile" name="Google Mobile Search crawler"
type="bot" />
<item token="Googlebot-Video" name="Googlebot-Video" type="bot" />
<item token="Googlebot-richsnippets" name="Googlebot Richsnippets"
type="bot" />
</root>
```

[parseUAs.lpr](#) Pascal (20,42 kByte) 15.06.2017 16:00

```

program parseUAs;
{$ifdef FPC}
 {$mode objfpc}
{$endif}
{$H+}

// interface
uses SysUtils, Classes, Dom, XMLRead;

type
 // UserAgent xml file info
 TUATypes = (uaBrowser, uaBot, uaCrawler, uaSpambot, uaCheck, uaProxy,
 uaDownload,
 uaPlugin, uaEngine, uaComponent, uaIgnore, uaOs,
 uaDistribution,
 uaCpu, uaDevice);
 TUAElement = record
 uaType: TUATypes;
 token, name, info: AnsiString;
 end;

 // UserAgent record
 TClientTypes = (ctBrowser, ctBot, ctCrawler, ctSpambot, ctCheck, ctProxy,
 ctDownload);
 TUserAgent = record
 ClientType: TClientTypes;
 ClientName, ClientVersion, ClientInfo, ClientRef,
 ClientBasedOn, ClientComponent, ClientPlugin, ClientLanguage:
 AnsiString;
 EngineName, EngineVersion: AnsiString;
 SystemName, SystemVersion, SystemDistribution: AnsiString;
 PlatformName, PlatformArchitecture: AnsiString;
 bot: boolean;
 remnant: AnsiString;
 end;
```

```

// UserAgent reader class
TUserAgentReader = class

private
  FElement: array of TUAElement;

public
  constructor Create(const AUserAgentXml: string); overload;
  destructor Destroy(); override;
  function GetUserAgentInfo(const AUserAgentString: string): TUserAgent;
  function GetEncodedUserAgentInfo(AUserAgentString: string): TUserAgent;
end;

// implementation

// reads the UserAgent xml file into the internal FElement array
constructor TUserAgentReader.Create(const AUserAgentXml: string);
var
  Doc: TXMLDocument;
  el: TDOMNode;
  i, a: integer;
begin
  inherited Create();
  if FileExists(AUserAgentXml) then
    begin
      try
        ReadXMLFile(Doc, AUserAgentXml);
        el := Doc.DocumentElement;
        with el.ChildNodes do
          begin
            SetLength(FElement, Count);
            for i:=0 to Count-1 do
              begin
                for a:=0 to Item[i].Attributes.Length-1 do
                  case Item[i].Attributes[a].NodeName of
                    'token': FElement[i].token := Item[i].Attributes[a].nodeValue;
                    'name': FElement[i].name := Item[i].Attributes[a].nodeValue;
                    'info': FElement[i].info := Item[i].Attributes[a].nodeValue;
                    'type': case Item[i].Attributes[a].nodeValue of
                      'browser': FElement[i].uaType := uaBrowser;
                      'bot': FElement[i].uaType := uaBot;
                      'spambot': FElement[i].uaType := uaSpambot;
                      'check': FElement[i].uaType := uaCheck;
                      'proxy': FElement[i].uaType := uaProxy;
                      'download': FElement[i].uaType := uaDownload;
                      'plugin': FElement[i].uaType := uaPlugin;
                      'engine': FElement[i].uaType := uaEngine;
                      'component': FElement[i].uaType := uaComponent;
                      'ignore': FElement[i].uaType := uaIgnore;
                      'os': FElement[i].uaType := uaOs;
                      'distribution': FElement[i].uaType :=

```



```

'su',
    'sv', 'sw', 'ta', 'te', 'tg', 'th', 'ti', 'tk', 'tl', 'tn', 'to',
'tr',
    'ts', 'tt', 'tw', 'ty', 'ug', 'uk', 'ur', 'uz', 've', 'vi', 'vo',
'wa',
    'wo', 'xh', 'yi', 'yo', 'za', 'zh', 'zu');
country: array[0..241] of string = (
    'AF', 'AL', 'DZ', 'AS', 'AD', 'AO', 'AI', 'AQ', 'AG', 'AR', 'AM',
'AW',
    'AU', 'AT', 'AZ', 'BS', 'BH', 'BD', 'BB', 'BY', 'BE', 'BZ', 'BJ',
'BM',
    'BT', 'BO', 'BA', 'BW', 'BV', 'BR', 'IO', 'VG', 'BN', 'BG', 'BF',
'BI',
    'KH', 'CM', 'CA', 'KV', 'KY', 'CF', 'TD', 'CL', 'CX', 'CC', 'CO',
'KM',
    'CG', 'CD', 'CK', 'CR', 'CI', 'HR', 'CU', 'CY', 'CZ', 'DK', 'DJ',
'DM',
    'DO', 'TP', 'EC', 'EG', 'SV', 'GQ', 'ER', 'EE', 'ET', 'FK', 'FO',
'FJ',
    'FI', 'FR', 'GF', 'PF', 'TF', 'GA', 'GM', 'GE', 'DE', 'GH', 'GI',
'GB',
    'GR', 'GL', 'GD', 'GP', 'GU', 'GT', 'GN', 'GW', 'GY', 'HT', 'HM',
'VA',
    'HN', 'HK', 'HU', 'IS', 'RE', 'IN', 'ID', 'IR', 'IQ', 'IE', 'IL',
'IT',
    'JM', 'JP', 'JO', 'KZ', 'KE', 'KI', 'XK', 'KW', 'KG', 'LA', 'LV',
'LB',
    'LS', 'LR', 'LY', 'LI', 'LT', 'LU', 'MO', 'MK', 'MG', 'MW', 'MY',
'MV',
    'ML', 'MT', 'MH', 'MQ', 'MR', 'MU', 'YT', 'MX', 'FM', 'MD', 'MC',
'MN',
    'ME', 'MS', 'MA', 'MZ', 'MM', 'NA', 'NR', 'NP', 'NL', 'AN', 'NC',
'NZ',
    'NI', 'NE', 'NG', 'NU', 'NF', 'KP', 'MP', 'NO', 'OM', 'PK', 'PW',
'PS',
    'PA', 'PG', 'PY', 'CN', 'PE', 'PH', 'PN', 'PL', 'PT', 'PR', 'QA',
'SM',
    'RO', 'RU', 'RW', 'KN', 'LC', 'PM', 'VC', 'WS', 'ST', 'SA', 'SN',
'RS',
    'SC', 'SL', 'SG', 'SK', 'SI', 'SB', 'SO', 'ZA', 'GS', 'KR', 'ES',
'LK',
    'SH', 'SD', 'SR', 'SJ', 'SZ', 'SE', 'CH', 'SY', 'TW', 'TJ', 'TZ',
'TH',
    'TG', 'TK', 'TO', 'TT', 'TN', 'TR', 'TM', 'TC', 'TV', 'UG', 'UA',
'AE',
    'UK', 'US', 'UY', 'UM', 'UZ', 'VU', 'VE', 'VN', 'VI', 'WF', 'EH',
'YE',
    'ZM', 'ZW');
function isLang(AString: string): boolean;
var i: integer;

```

```

begin
  for i := Low(lang) to High(lang) do
    if lang[i] = AString then
      exit(true);
    result := false;
  end;
  function isCountry(AString: string): boolean;
  var i: integer;
  begin
    if (AString[1] in ['A'..'Z']) then
      begin
        for i := Low(country) to High(country) do
          if country[i] = AString then
            exit(true)
      end
    else
      for i := Low(country) to High(country) do
        if lowercase(country[i]) = AString then
          exit(true);
    result := false;
  end;
  function isLangCountry(AString: string): boolean;
  begin
    if (length(AString) = 2) or ((length(AString) = 5) and (AString[3] in
['-', '_'])) then
      begin
        if (length(AString) = 5) and not isCountry(copy(AString, 4, 2)) then
          exit(false);
        result := isLang(copy(AString, 1, 2));
      end
    else
      result := false;
  end;

var
  bVersion, bRv, bName: boolean;
  sName, sVersion: string;

  procedure checkPart(const last: boolean = false);
  var
    bFound: boolean;
    i, l: integer;
    s: string;
  begin
    sName := trim(sName);
    sVersion := trim(sVersion);
    if sName <> '' then
      with result do
        begin
          bFound := false;

```

```

for i := Low(FElement) to High(FElement) do
begin
  s := FElement[i].token;
  l := length(s);
  if (s[l] in ['/','*']) and (pos(copy(s, 1, l-1), sName)=1) then
    begin
      // "/" = a Version is following
      // "*" = the following is not of interest
      bFound := true;
      if (sVersion = '') and (s[l] = '/') then
        begin
          sVersion := copy(sName, 1);
          if pos(' ', sVersion) > 0 then
            sVersion := copy(sVersion, 1, pos(' ', sVersion)-1);
        end;
        break;
      end;
    else if (s = sName) then
      begin
        // exact match
        bFound := true;
        break;
      end;
    end;
  if bFound then
    begin
      // a match is found in storage
      case FElement[i].uaType of
        uaBrowser:
          // it's a browser
          if not bName then
            begin
              ClientType := ctBrowser;
              ClientName := ClientName + BoolToStr(ClientName<>' ', ' ', '') +
FElement[i].name;
              if (FElement[i].name <> 'Mobile') then
                bName := true;
              if not bVersion then
                ClientVersion := sVersion;
              ClientInfo := ClientInfo + BoolToStr(ClientInfo<>' ', ' ', '') +
FElement[i].info;
            end;
          uaBot, uaCrawler, uaSpambot, uaCheck, uaProxy, uaDownload:
            // it's some kind of bot
            begin
              case FElement[i].uaType of
                uaBot: ClientType := ctBot;
                uaCrawler: ClientType := ctCrawler;
                uaSpambot: ClientType := ctSpambot;
                uaCheck: ClientType := ctCheck;

```

```

uaProxy: ClientType := ctProxy;
uaDownload: ClientType := ctDownload;
end;
ClientName := FElement[i].name;
bName := true;
if sVersion <> '' then
    ClientVersion := sVersion;
    bot := true;
end;
uaEngine:
// it's a engine
if ((sVersion <> '') or bRv) then
begin
    EngineName := FElement[i].name;
    if not bRv then
        EngineVersion := sVersion;
    end;
uaPlugin:
// it's some kind of plugin
ClientPlugin := ClientPlugin + BoolToStr(ClientPlugin<>' ', ',',
', '') + FElement[i].name + BoolToStr(sVersion<>' ', '/'+ sVersion, '');
uaComponent:
// it's some kind of component
ClientComponent := ClientComponent +
BoolToStr(ClientComponent<>' ', ', ', ', ') + FElement[i].name +
BoolToStr(sVersion<>' ', '/'+ sVersion, '');
uaOs:
// it's a operating system
if (pos('Windows', FElement[i].name)=1) then
begin
    SystemName := 'Windows';
    SystemVersion := copy(FElement[i].name, 9);
end
else
begin
    if (FElement[i].name = 'Linux') and (sVersion <> '') then
begin
        PlatformArchitecture := sVersion;
        sVersion := '';
    end;
    SystemName := BoolToStr(SystemName<>' ', SystemName + ' ', '');
+ FElement[i].name;
    if sVersion <> '' then
begin
        if pos('_', sVersion) > 0 then
            sVersion := StringReplace(sVersion, '_', '.', ,
[rfReplaceAll]);
        SystemVersion := sVersion;
    end;
end;

```

```

uaDistribution:
    // it's a operating system distribution
    SystemDistribution := FElement[i].name + BoolToStr(sVersion<>'',
'/' + sVersion, '');

uaCpu:
    // it's some kind of processor
    PlatformArchitecture := FElement[i].name +
BoolToStr(sVersion<>'', '/' + sVersion, '');

uaDevice:
    // it's some device info
    PlatformName := FElement[i].name;
end
end
else if sName = 'rv' then
begin
    EngineVersion := sVersion;
    bRv := true;
end
else if (sName = 'Version') and (sVersion <> '') then
begin
    if not bot then
begin
    ClientVersion := sVersion;
    bVersion := true;
end;
end
else if (pos('http:', sName)>0) or (pos('https:', sName)>0) or
(pos('@', sName)>0) then
    ClientRef := sName
else if pos('Win64', sName)>0 then
    SystemVersion := SystemVersion + ' 64bit'
else if pos('WOW64', sName)>0 then
begin
    SystemVersion := SystemVersion + ' 32bit';
    PlatformArchitecture := 'x86_64';
end
else if isLangCountry(sName) then
    ClientLanguage := sName
{ else if last and (ClientName = '') then
begin // catch bots
    ClientName := sName;
    if sVersion <> '' then
        ClientVersion := sVersion;
    bot := true;
end }
else if (pos('crawler', lowercase(sName))>0) or
        (pos('spider', lowercase(sName))>0) or
        (pos('bot', lowercase(sName))>0) or
        (pos('Bot', lowercase(sName))>0) then
    remnant := remnant + BoolToStr(remnant<>' ', ' ', '') + '|'+<item

```

```

token=""+sName+" name='"+sName+"' type="bot" />' + '|'
else
    remnant := remnant + BoolToStr(remnant<>', ', ', '') +
'|'+sName+BoolToStr(sVersion<>', '/'+sVersion, '') + '|';
end;
sName := '';
sVersion := '';
end;

type TStates = (STATE_UNDEFINED, STATE_NAME, STATE_VERSION, STATE_COMMENT,
STATE_COMMENT_VERSION, STATE_LANGUAGE);
var
state, fallbackState: TStates;
i, len: integer;
c: char;
begin
Finalize(result);
FillChar(result, SizeOf(result), 0);
result.bot := false;

// check if it can be a user-agent
if (AUserAgentString[1] in ['A'..'Z', 'a'..'z', '[']) then
begin
// init start values
bVersion := false;
bRv := false;
bName := false;
sName := '';
sVersion := '';
state := STATE_UNDEFINED;
fallbackState := STATE_UNDEFINED;
len := Length(AUserAgentString);
for i:=1 to len do
begin
c := AUserAgentString[i];
// the state is undefined after each part and at the beginning
if (state = STATE_UNDEFINED) then
begin
if (c = '(') then
// start of comment
state := STATE_COMMENT
else if (c = '[') then
// start of old language part
state := STATE_LANGUAGE
else if not (c in [' ', '+', ')', ';']) then // ignores this at the beginnig of a new part
begin
// start of a name part
state := STATE_NAME;
sName := c;

```

```

    end
end
else if (state = STATE_NAME) then
begin
  if (c = '/') then
    // start of the version part
    state := STATE_VERSION
  else if (c = '+') then
begin
  // something new starts
  checkPart();
  state := STATE_UNDEFINED
end
else if (c = '(') then
begin
  // start of comment
  checkPart();
  state := STATE_COMMENT
end
else
  // go on writing the name
  sName := sName + c;
end
else if (state = STATE_VERSION) then
begin
  if (c = ' ') then
begin
  // something new starts
  state := STATE_UNDEFINED;
  checkPart();
end
else if (c = '/') then
begin
  // the first slash was part of the name (fix)
  sName := sName + '/' + sVersion;
  sVersion := '';
end
else
  // go on writing the version
  sVersion := sVersion + c;
end
else if (state in [STATE_COMMENT, STATE_COMMENT_VERSION]) then
begin
  if (c = ')') then
begin
  // the comment ends, something new starts
  checkPart();
  state := fallbackState;
end
else if (c = '(') then

```

```

begin
    // there is a comment inner comment
    checkPart();
    fallbackState := STATE_COMMENT;
    state := STATE_COMMENT;
end
else if (c in [';', ',',']) then
begin
    // next comment
    checkPart();
    state := STATE_COMMENT;
end
else if (c in ['+', ' ']) and (sName = '') then
    // ignore + and space of the beginning of a comment
else if (c = '/') and (pos('http:', sName)=0) and (pos('https:', sName)=0) then
    // start with version in comment part
    state := STATE_COMMENT_VERSION
else if (c = ':') and (sName = 'rv') then
    // rv is a special case it is separated by a colon
    // start with version in comment part
    state := STATE_COMMENT_VERSION
else
begin
    if (state = STATE_COMMENT_VERSION) then
        // go on writing the comment version
        sVersion := sVersion + c
    else
        // go on writing the comment name
        sName := sName + c;
end;
end
else if (state = STATE_LANGUAGE) then
begin
    if (c = ']') then
        begin
            checkPart();
            state := STATE_UNDEFINED
        end
    else
        sName := sName + c;
    end;
end;
// analyse the last part
checkPart(true);
end;
if (result.ClientName = '') then
    result.ClientName := 'unknown';
end;

```

```

function TUserAgentReader.GetEncodedUserInfo(AUserAgentString: string): 
TUserAgent;
begin
  AUserAgentString := StringReplace(AUserAgentString, '++', ' '#9,
[rfReplaceAll]);
  AUserAgentString := StringReplace(AUserAgentString, '+(+http', ' '
(' #9' http', [rfReplaceAll]));
  AUserAgentString := StringReplace(AUserAgentString, '+', ' ', 
[rfReplaceAll]);
  AUserAgentString := StringReplace(AUserAgentString, '#9, '+,
[rfReplaceAll]);
  result := GetUserInfo(AUserAgentString);
end;

function Dump(const AUserAgent: TUserAgent): string;
var s: string;
begin
  with AUserAgent do
  begin
    result := 'client' +
      BoolToStr(bot, '(bot)', '') + ': ' + ClientName +
      BoolToStr(ClientVersion <> '', ' ' + ClientVersion, '') +
      BoolToStr(ClientInfo <> '', ' (' + ClientInfo + ')', '') + #$0A;

    s := '';
    if (ClientRef <> '') then
      s := s + BoolToStr(s = '', ' ', '; ') + 'ref: ' + ClientRef;
    if (ClientBasedOn <> '') then
      s := s + BoolToStr(s = '', ' ', '; ') + 'based on: ' +
ClientBasedOn;
    if (ClientComponent <> '') then
      s := s + BoolToStr(s = '', ' ', '; ') + 'component: ' +
ClientComponent;
    if (ClientPlugin <> '') then
      s := s + BoolToStr(s = '', ' ', '; ') + 'plugin: ' + ClientPlugin;
    if (ClientLanguage <> '') then
      s := s + BoolToStr(s = '', ' ', '; ') + 'language: ' +
ClientLanguage;
    if s <> '' then
      result := result + s + #$0A;

    if (EngineName <> '') then
      result := result + 'engine: ' +
        EngineName + BoolToStr(EngineVersion <> '', ' ' + EngineVersion, '') +
+ #$0A;

    if (PlatformName <> '') or (PlatformArchitecture <> '') then
      result := result + 'platform:' +
        BoolToStr(PlatformName <> '', ' ' + PlatformName, '') +
        BoolToStr(PlatformArchitecture <> '', ' ' + PlatformArchitecture,

```

```

') + #$0A;

    if (SystemDistribution<>'') or (SystemName <> '') or (SystemVersion <>
'') then
        result := result + 'system:' +
            BoolToStr(SystemDistribution <> '', ' ' + SystemDistribution, '') +
            BoolToStr(SystemName <> '', ' ' + SystemName, '') +
            BoolToStr(SystemVersion <> '', ' ' + SystemVersion, '') + #$0A;
        if (remnant <> '') then
            result := result + 'remnant: ' + remnant + #$0A;
    end;
end;

var
    i: integer;
    sUserAgent: string;
    UserAgent: TUserAgent;
    uaReader: TUserAgentReader;
    f: TextFile;
    StartTime, EndTime: QWord;
begin
    StartTime := GetTickCount64();
    uaReader := TUserAgentReader.Create(ExtractFilePath(ParamStr(0)) +
'ua.xml');
    with TStringList.Create() do
        try
            LoadFromFile(ExtractFilePath(ParamStr(0)) + 'useragents.txt');
            AssignFile(f, ExtractFilePath(ParamStr(0)) + 'useragents~my.txt');
            rewrite(f);
            for i:=0 to Count-1 do
                begin
                    sUserAgent := Strings[i];
                    UserAgent := uaReader.GetUserInfo(sUserAgent);
                    writeln(f, '*' + sUserAgent);
                    writeln(f, Dump(UserAgent));
                end;
            CloseFile(f);
        finally
            free;
        end;
    uaReader.Free;
    EndTime := GetTickCount64();
    WriteLn('TickCount: ' + IntToStr(EndTime - StartTime) + ' ms');
    ReadLn();
end.

```

Einfach im selben Verzeichnis starten, dann wird eine Datei useragents~my.txt angelegt es ist die Aufschlüsselung der useragents.txt Datei die wir im Schritt 1. erzeugt haben, dieser Vorgang dauert nur wenige Sekunden.

 [parseUAs.zip](#) kompilierte Win32 Exe-Datei

(129,28 kByte) 30.12.2018 21:29

Autor: [Udo Schmal](#), veröffentlicht: 02.06.2017, letzte Änderung: 29.08.2024

[© Copyright 2024 Udo Schmal](#)