GMT - Greenwich Mean Time

Zum generieren des HTML-Headers sind für die Werte Last-Modified und Date Funktionen zur Ausgabe des Datum/Zeit Wertes im GMT-Format erforderlich.

Des weiteren werden in der kompletten Datenkommunikation immer GMT Zeitangaben benötigt um eine global über die Zeit-Zonen hinaus gültige Angabe zu haben, lokale Zeitangaben sind für den einzelnen Betrachter noch relevant, aber in der Zeit der Globalisierung gibt es sonst keine genaue Vergleichsmöglichkeit. So werden diese Funktionen auch zum Füllen des Meta-Wertes Date im head-Bereich des HTML-Dokumentes benötigt und in Atom-, RSS- Feeds, etc.

gmtutils.pas Pascal (13,02 kByte) 23.02.2014 13:59

```
//
******************************************************************************

//  Title.............. :  GMT Utils Unit
//
//  Modulname ......... :  GMTUtils.pas
//  Type .............. :  Unit
//  Author ............ :  Udo Schmal
//  Development Status  :  15.02.2014
//  Operating System .. :  Win32/Win64
//  IDE ............... :  Lazarus
//
******************************************************************************

unit GMTUtils;
{$mode objfpc} {$H+}

interface

uses
  {$ifdef windows}Windows{$endif}, Classes, SysUtils, DateUtils;

type
  TFileTimes = set of (ftCreation, ftLastAccess, ftLastWrite);

const
  tsRFC     = 'ddd, dd mmm yyyy hh:nn:ss "GMT"';             // Mon, 15 Aug
2005  15:52:01 GMT
  tsRFC3339 = 'yyyy"-"mm"-"dd"T"hh":"nn":"ss"+00:00"';       // 2005-08-
15T15:52:01+00:00
  tsAtom    = 'yyyy"-"mm"-"dd"T"hh":"nn":"ss"Z"';           //  tsAtom    =
tsRFC3339;
  tsW3C     = tsRFC3339;
  tsiCal    = 'yyyymmdd"T"hhnnss"Z"';                       //
20050815T155201Z
  tsGER     = 'dd"."mm"."yyyy hh":"mm":"ss"."zzz';          // 15.08.2005
15:52:01.123
  tsISO8601 = 'yyyy"-"mm"-"dd"T"hh":"nn":"ss"+0000"';       // 2005-08-
```

```
15T15:52:01+0000
  tsRFC822  = 'ddd", "dd" "mmm" "yy" "hh":"nn":"ss" +0000"'; // Mon, 15 Aug
05 15:52:01 +0000
  tsRFC1036 = tsRFC822;
  tsRFC850  = 'dddd", "dd"-"mmm"-"yy" "hh":"nn":"ss" UTC"';  // Monday, 15-
Aug-05 15:52:01 UTC
  tsCookie  = tsRFC850;
  tsRFC1123 = 'ddd", "dd" "mmm" "yyyy" "hh":"nn":"ss" +0000"'; // Mon, 15
Aug 2005 15:52:01 +0000
  tsRFC2822 = tsRFC1123;
  tsRSS     = tsRFC1123;
  tsLog     = 'yyyy"-"mm"-"dd" "hh":"nn":"ss';

  GD_LCID = 2057;
  DE_LCID = 1031;

function DateTimeToString(ADateTime: TDateTime; const frmtstr: string =
tsRFC; const LCID: integer = GD_LCID): string;
function StringToDateTime(ADateTime: string; const frmtstr: string = tsRFC;
const LCID: integer = GD_LCID): TDateTime;

function NowGMT(): TDateTime;
function NowGMTStr(const frmtstr: string = tsRFC): string;

function LocalTimeToGMT(const ALocalDateTime: TDateTime): TDateTime;
function GMTToLocalTime(const AGMTTime: TDateTime): TDateTime;
function LocalTimeToGMTStr(const ALocalDateTime: TDateTime; const frmtstr:
string = tsRFC; const LCID: integer = GD_LCID): string;

function FileTimeGMT(SearchRec: TSearchRec): TDateTime; overload;
function FileTimeGMT(const AFileTime: TFileTime): TDateTime; overload;
function FileTimeGMT(AFilename: string): TDateTime; overload;
function FileTimeGMTStr(const AFilename: string; const frmtstr: string =
tsRFC; const LCID: integer = GD_LCID): string;

function FileTimeCopy(const AFrom, ATo: string): boolean;

function UnixDateTimeToDateTime(const AUnixDateTime: LongInt): TDateTime;
function UnixTimeToDateTime(const UnixTime: integer): TDateTime;
function DateTimeToUnixDateTime(const ADateTime: TDateTIme): Longint;
function DateTimeToUnixTime(const DateTime: TDateTime): integer;

implementation

function DateTimeToString(ADateTime: TDateTime; const frmtstr: string =
tsRFC; const LCID: integer = GD_LCID): string;
var fs: TFormatSettings;
begin
  GetLocaleFormatSettings(LCID, fs);
  result := FormatDateTime(frmtstr, ADateTime, fs);
```

```
end;

function StringToDateTime(ADateTime: string; const frmtstr: string = tsRFC;
const LCID: integer = GD_LCID): TDateTime;
var fs: TFormatSettings;
begin
  GetLocaleFormatSettings(LCID, fs);
  if ADateTime='' then
  begin
    ADateTime := '01.01.1970 00:00:00';
    fs.DateSeparator := '.';
    fs.TimeSeparator := ':';
    fs.DecimalSeparator := '.';
  end;
  if frmtstr = tsRFC then
  begin
    fs.ShortDateFormat := 'ddd", "dd" "mmm" "yyyy';
    ADateTime := StringReplace(ADateTime, 'T', ' ', []);
    fs.LongTimeFormat := 'hh":"nn":"ss';
    ADateTime := StringReplace(ADateTime, ' GMT', '', []);
    fs.DateSeparator := ' ';
  end
  else if frmtstr = tsRFC822 then
  begin
    fs.ShortDateFormat := 'ddd", "dd" "mmm" "yy';
    fs.LongTimeFormat := 'hh":"nn":"ss';
    ADateTime := StringReplace(ADateTime, ' +0000', '', []);
    fs.DateSeparator := ' ';
  end
  else if frmtstr = tsRFC850 then
  begin
    fs.ShortDateFormat := 'dddd", "dd"-"mmm"-"yy';
    fs.LongTimeFormat := 'hh":"nn":"ss';
    ADateTime := StringReplace(ADateTime, ' UTC', '', []);
    fs.DateSeparator := '-';
  end
  else if frmtstr = tsRFC1123 then
  begin
    fs.ShortDateFormat := 'ddd", "dd" "mmm" "yyyy';
    fs.LongTimeFormat := 'hh":"nn":"ss';
    ADateTime := StringReplace(ADateTime, ' +0000', '', []);
    fs.DateSeparator := ' ';
  end
  else if frmtstr = tsRFC3339 then
  begin
    fs.ShortDateFormat := 'yyyy"-"mm"-"dd';
    ADateTime := StringReplace(ADateTime, 'T', ' ', []);
    fs.LongTimeFormat := 'hh":"nn":"ss';
    ADateTime := StringReplace(ADateTime, '+00:00', '', []);
    fs.DateSeparator := '-';
```

```
  end
  else if frmtstr = tsISO8601 then
  begin
    fs.ShortDateFormat := 'yyyy"-"mm"-"dd';
    ADateTime := StringReplace(ADateTime, 'T', ' ', []);
    fs.LongTimeFormat := 'hh":"nn":"ss';
    ADateTime := StringReplace(ADateTime, '+0000', '', []);
    fs.DateSeparator := '-';
  end
  else if frmtstr = tsGER then
  begin
    fs.ShortDateFormat := 'dd"."mm"."yyyy';
    fs.LongTimeFormat := 'hh":"nn":"ss"."zzz';
    fs.DateSeparator := '.';
    fs.TimeSeparator := ':';
    fs.DecimalSeparator := '.';
  end;
  result := StrToDateTime(ADateTime, fs);
end;

function NowGMT(): TDateTime;
{$ifdef windows}
var SystemTime: TSystemTime;
begin
  GetSystemTime(SystemTime);
  result := SystemTimeToDateTime(SystemTime);
end;
{$else unix}
var
  TimeVal: TTimeVal;
  TimeZone: PTimeZone;
  a: Double;
begin
  TimeZone := nil;
  fpGetTimeOfDay(@TimeVal, TimeZone);
  // Convert to milliseconds
  a := (TimeVal.tv_sec * 1000.0) + (TimeVal.tv_usec / 1000.0);
  result := (a / MSecsPerDay) + UnixDateDelta;
end;
{$endif}

function NowGMTStr(const frmtstr: string = tsRFC): string;
begin
  result := DateTimeToString(NowGMT(), frmtstr);
end;

{$ifdef windows}
function CompareSysTime(st1, st2: TSystemTime): integer;
begin
  if st1.wMonth < st2.wMonth then result := -1
```

```
  else if st1.wMonth > st2.wMonth then result := 1
  else if st1.wDayOfWeek < st2.wDayOfWeek then result := -1
  else if st1.wDayOfWeek > st2.wDayOfWeek then result := 1
  else if st1.wDay < st2.wDay then result := -1
  else if st1.wDay > st2.wDay then result := 1
  else if st1.wHour < st2.wHour then result := -1
  else if st1.wHour > st2.wHour then result := 1
  else if st1.wMinute < st2.wMinute then result := -1
  else if st1.wMinute > st2.wMinute then result := 1
  else if st1.wSecond < st2.wSecond then result := -1
  else if st1.wSecond > st2.wSecond then result := 1
  else if st1.wMilliseconds < st2.wMilliseconds then result := -1
  else if st1.wMilliseconds > st2.wMilliseconds then result := 1
  else result := 0;
end;
{$endif}
function GetTZInfoForDateTime(const ADateTime: TDateTime): integer;
var
{$ifdef windows}
  SystemTime: TSystemTime;
  TZInfo: TTimeZoneInformation;
{$endif}
begin
{$ifdef windows}
  GetTimeZoneInformation(TZInfo);
  DateTimeToSystemTime(ADateTime, SystemTime);
  if (CompareSysTime(SystemTime, TZInfo.DayLightDate)=1) and
     (CompareSysTime(SystemTime, TZInfo.StandardDate)=-1) then
    result := TZInfo.Bias + TZInfo.DaylightBias
  else
    result := TZInfo.Bias + TZInfo.StandardBias;
{$else unix}
  result := -TZseconds div 60;
{$endif}
end;

function LocalTimeToGMT(const ALocalDateTime: TDateTime): TDateTime;
var TZOffset: integer;
begin
  result := ALocalDateTime;
  TZOffset := GetTZInfoForDateTime(ALocalDateTime);
  if (TZOffset > 0) then
// Time zones west of Greenwich.
    result := ALocalDateTime + EncodeTime(TZOffset div 60, TZOffset mod 60,
0, 0)
  else if (TZOffset = 0) then
// Time Zone = Greenwich.
    result := ALocalDateTime
  else if (TZOffset < 0) then
// Time zones east of Greenwich.
```

```
    result := ALocalDateTime - EncodeTime(Abs(TZOffset) div 60,
Abs(TZOffset) mod 60, 0, 0);
end;

function GMTToLocalTime(const AGMTTime: TDateTime): TDateTime;
var TZOffset: Integer;
begin
  result := AGMTTime;
  TZOffset := GetTZInfoForDateTime(AGMTTime);
  if (TZOffset > 0) then
// Time zones west of Greenwich.
    result := AGMTTime - EncodeTime(TZOffset div 60, TZOffset mod 60, 0, 0)
  else if (TZOffset = 0) then
// Time Zone = Greenwich.
    result := AGMTTime
  else if (TZOffset < 0) then
// Time zones east of Greenwich.
    result := AGMTTime + EncodeTime(Abs(TZOffset) div 60, Abs(TZOffset) mod
60, 0, 0);
end;


function LocalTimeToGMTStr(const ALocalDateTime: TDateTime; const frmtstr:
string = tsRFC; const LCID: integer = GD_LCID): string;
begin
  result := DateTimeToString(LocalTimeToGMT(ALocalDateTime), frmtstr, LCID);
end;

function FileTimeGMT(SearchRec: TSearchRec): TDateTime;
var
{$ifdef windows}
  SystemFileTime: TSystemTime;
{$else unix}
  TimeVal: TTimeVal;
  TimeZone: TTimeZone;
{$endif}
begin
  result := 0.0;
{$ifdef windows}
  {$WARNINGS OFF}
  if (SearchRec.FindData.dwFileAttributes and faDirectory) = 0 then
    if FileTimeToSystemTime(SearchRec.FindData.ftLastWriteTime,
SystemFileTime) then
      result := EncodeDate (SystemFileTime.wYear, SystemFileTime.wMonth,
SystemFileTime.wDay)
              + EncodeTime (SystemFileTime.wHour, SystemFileTime.wMinute,
SystemFileTime.wSecond, SystemFileTime.wMilliseconds);
  {$WARNINGS ON}
{$else unix}
  if SearchRec.Attr and faDirectory = 0 then
```

```
  begin
    result := FileDateToDateTime(SearchRec.Time);
    fpGetTimeOfDay(@TimeVal, @TimeZone);
    result := result + TimeZone.tz_minuteswest / (60 * 24);
  end;
{$endif}
end;

function FileTimeGMT(const AFileTime: TFileTime): TDateTime;
var
{$ifdef windows}
  SystemFileTime: TSystemTime;
{$else unix}
  TimeVal: TTimeVal;
  TimeZone: TTimeZone;
{$endif}
begin
  result := 0.0;
{$ifdef windows}
  {$WARNINGS OFF}
  if FileTimeToSystemTime(AFileTime, SystemFileTime) then
    result := EncodeDate (SystemFileTime.wYear, SystemFileTime.wMonth,
SystemFileTime.wDay)
              + EncodeTime (SystemFileTime.wHour, SystemFileTime.wMinute,
SystemFileTime.wSecond, SystemFileTime.wMilliseconds);
  {$WARNINGS ON}
{$else unix}
  if SearchRec.Attr and faDirectory = 0 then
  begin
    result := FileDateToDateTime(AFileTime);
    fpGetTimeOfDay(@TimeVal, @TimeZone);
    result := result + TimeZone.tz_minuteswest / (60 * 24);
  end;
{$endif}
end;

function FileTimeGMT(AFilename: string): TDateTime;
var sr: TSearchRec;
begin
  result := 0.0;
  if FindFirst (AFilename, faAnyFile, sr) = 0 then
    result := FileTimeGMT(sr);
  FindClose(sr);
end;

function FileTimeGMTStr(const AFilename: string; const frmtstr: string =
tsRFC; const LCID: integer = GD_LCID): string;
begin
  result := DateTimeToString(FileTimeGMT(AFilename), frmtstr, LCID);
end;
```

```
function FileTimeCopy(const AFrom, ATo: string): boolean;
var
{$ifdef windows}
  fh : THandle;
  fCreationTime, fLastAccessTime, fLastWriteTime: TFileTime;
{$else}
  fa : Longint;
{$endif}
begin
  result := false;
  if FileExists(AFrom) and FileExists(ATo) then
  begin
{$ifdef windows}
    fh := FileOpen(AFrom, fmOpenRead or fmShareDenyNone);
    if fh > 0 then // <> INVALID_HANDLE_VALUE then
    try
      result := GetFileTime(fh, @fCreationTime, @fLastAccessTime,
@fLastWriteTime);
    finally
      FileClose(fh);
    end;
    fh := FileOpen(ATo, fmOpenWrite);
    if fh > 0 then
    try
      result := SetFileTime(fh, @fCreationTime, @fLastAccessTime,
@fLastWriteTime);
    finally
      FileClose(fh);
    end;
{$else}
    fa := FileAge(AFrom);
    if fa<>-1 then
      result := FileSetDate(ATo, FileAge(AFrom)) = 0
    else
      result := false;
{$endif}
  end;
end;

const UnixStartDate: TDateTime = 25569.0; // UnixStartDate to TDateTime of
01/01/1970

function UnixDateTimeToDateTime(const AUnixDateTime: LongInt): TDateTime;
begin
  Result := UnixStartDate + (AUnixDateTime / 86400); {86400=No. of secs. per
day}
end;

function UnixTimeToDateTime(const UnixTime: integer): TDateTime;
```

```
var
  FileTime: TFileTime;
  SystemTime: TSystemTime;
  I: Int64;
begin
  // first convert unix time to a Win32 file time
  I := Int64(UnixTime) * Int64(10000000) + 116444736000000000;
  FileTime.dwLowDateTime := DWORD(I);
  FileTime.dwHighDateTime := I shr 32;
  // now convert to system time
  FileTimeToSystemTime(FileTime, SystemTime);
  // and finally convert the system time to TDateTime
  Result := SystemTimeToDateTime(SystemTime);
end;

function DateTimeToUnixDateTime(const ADateTime: TDateTIme): Longint;
begin
  Result := Round((ADateTime - UnixStartDate) * 86400); {86400=No. of secs.
per  day}
end;

function DateTimeToUnixTime(const DateTime: TDateTime): integer;
var
  FileTime: TFileTime;
  SystemTime: TSystemTime;
  I: Int64;
begin
  // first convert datetime to Win32 file time
  DateTimeToSystemTime(DateTime, SystemTime);
  SystemTimeToFileTime(SystemTime, FileTime);
  // simple maths to go from Win32 time to Unix time
  I := Int64(FileTime.dwHighDateTime) shl 32 + FileTime.dwLowDateTime;
  Result := (I - 116444736000000000) div Int64(10000000);
end;

end.
```

Autor: Udo Schmal, veröffentlicht: 12.06.2012, letzte Änderung: 29.08.2024

© Copyright 2024 Udo Schmal