Recycle IIS Application Pool

Kennt Ihr das Problem der Server (IIS) hängt immer wieder aber Ihr seid nicht in der Lage den Fehler zu finden, denn der Server hängt ja und kommt gar nicht dazu einen Eintrag ins Logfile zu schreiben und so findet Ihr nicht mal das fehlerhafte Script, geschweige denn die fehlerhaften Übergabeparameter.

Aber das ist nicht mal das Schlimmste, denn durch das zuvor geschriebene Filter seit Ihr nun in der Lage den Fehler zu finden, das Schlimmste ist der Server muss neu gestartet werden und das an Besten automatisch, et voilà! Unter Windows 7 existiert schon ein Kommando appcmd recycle apppool (https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc770764(v=ws.10)) (z.B.: appcmd recycle apppool /apppool.name:MyWebpagePool) nur bei älteren Servern wir was Eigenes benötigt, ich setze jetzt mal voraus das von Mircosoft entwickelte Programm erfüllt seine Bestimmung, wie z.B. das folgende Programm RecycleAppPool.zip. (87,3 kByte) 30.12.2018 21:29

RecycleAppPool.lpr Pascal (4,59 kByte) 25.02.2012 17:28

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

//  Title.............. :  Recycle IIS Application Pool
//
//  Modulname ......... :  RecycleAppPool.lpr (Console)
//  Type .............. :  Lazarus Project File
//  Author ............ :  Udo Schmal
//  Development Status  :  13.12.2011
//  Operating System .. :  Win32
//  IDE ............... :  Lazarus
//
******************************************************************************

program RecycleAppPool;
{$mode objfpc}{$H+}
{$R *.res}

uses SysUtils, ActiveX, ComObj{, Variants , WbemScripting_TLB};

{type
  ISWbemObjectSet = interface(IDispatch)
    ['{76A6415F-CB41-11D1-8B02-00600806D9B6}']
    function Get__NewEnum: IUnknown; safecall;
    function Item(const strObjectPath: WideString; iFlags: Integer):
ISWbemObject; safecall;
    function Get_Count: Integer; safecall;
    function Get_Security_: ISWbemSecurity; safecall;
    function ItemIndex(lIndex: Integer): ISWbemObject; safecall;
    property _NewEnum: IUnknown read Get__NewEnum;
    property Count: Integer read Get_Count;
    property Security: ISWbemSecurity read Get_Security_;
  end;}

var LogFile: string;
```

```
procedure Log(const sLine: string);
var f: textfile;
begin
{$I-}
  AssignFile(f, LogFile);
  if FileExists(LogFile) then
    Append(f)
  else
    ReWrite(f);
  WriteLn(f, FormatDateTime('dd"."mm"."yyyy hh:nn:ss:zzz"|"', Now) + sLine);
  CloseFile(f);
  IOResult;
{$I+}
end;
{
procedure GetListWMINameSpaces(const RootNameSpace: WideString);
var
  objSWbemLocator : OleVariant;
  objWMIService   : OleVariant;
  colItems        : OleVariant;
  colItem         : Variant;
  oEnum           : IEnumvariant;
  iValue          : LongWord;
  sValue          : String;
begin
  try
    objSWbemLocator :=
CreateOleObject(WideString('WbemScripting.SWbemLocator'));
    objSWbemLocator.Security_.ImpersonationLevel := 3;
//WBemImpersonationLevelImpersonate
    objSWbemLocator.Security_.AuthenticationLevel := 6;
//WBemAuthenticationLevelPktPrivacy
    objWMIService   := objSWbemLocator.ConnectServer(WideString('.'),
RootNameSpace, '', '');
    if RootNameSpace=WideString('root\WebAdministration') then
    begin
      colItems := objWMIService.ExecQuery(WideString('SELECT * FROM Site'))
      oEnum    := IUnknown(colItems._NewEnum) as ISWbemObject;
    end
    else
    begin
      colItems := objWMIService.InstancesOf(WideString('__NAMESPACE'));
      oEnum    := IUnknown(colItems._NewEnum) as IEnumVariant;
    end;
    while oEnum.Next(1, colItem, @iValue) = 0 do
    begin
      sValue := VarToStr(colItem.Name);
      if RootNameSpace=WideString('root\WebAdministration') then
        WriteLog(etInfo, 'GetListWMINameSpaces.Site: ' + sValue)
      else
```

```
      begin
        colItem := Unassigned;
        //List.Add(RootNameSpace+'\'+sValue);
        colItem := Unassigned;
        //List.Add(RootNameSpace+'\'+sValue);
        WriteLog(etInfo, 'GetListWMINameSpaces: ' +
RootNameSpace+'\'+sValue);
        GetListWMINameSpaces(RootNameSpace+'\'+sValue);
      end;
    end;
  except
    WriteLog(etError, 'GetListWMINameSpaces in ' + RootNameSpace);
  end;
end;
}

function ParentDir(const sPath: string): string;
begin
  result := Copy(sPath, 1, LastDelimiter(':\',
copy(sPath,1,length(sPath)-1)));
end;

var
  Locator, Provider, Pool: Variant;
  strPoolName: string;
begin
  LogFile := ParentDir(ExtractFilePath(ParamStr(0))) + 'logging\' +
ChangeFileExt(ExtractFileName(ParamStr(0)), '.log');
  strPoolName := ParamStr(1);
  Log('RecycleAppPool: ' + strPoolName);
  CoInitialize(NIL);
  try
//    GetListWMINameSpaces(WideString('root'));
//    GetListWMINameSpaces(WideString('root/MicrosoftIISv2'));
//    GetListWMINameSpaces(WideString('root\WebAdministration'));

    // Initializes the WMI Locator object
    Locator := CreateOleObject('WbemScripting.SWbemLocator'); //
%WINDIR%\system32\wbem\wbemdisp.dll
    Locator.Security_.ImpersonationLevel := 3;
//WBemImpersonationLevelImpersonate
    Locator.Security_.AuthenticationLevel := 6;
//WBemAuthenticationLevelPktPrivacy
    // WMI Connect
    Provider := Locator.ConnectServer(WideString('.'),
WideString('root/MicrosoftIISv2')); //WMI_NAMESPACE
    Pool := Provider.Get(WideString('IIsApplicationPool=''W3SVC/AppPools/' +
strPoolName + ''''));
    Pool.Recycle;
    Log('Recycled Application Pool: ' + strPoolName);
```

```
  except
    on E: Exception do
      Log('Error in: RecycleAppPool.exe ' + strPoolName + #13#10 +
          'Error ClassName: ' + E.ClassName + #13#10 +
          'Error Message: ' + E.Message);
  end;
  CoUnInitialize;
end.
```

Autor: Udo Schmal, veröffentlicht: 11.07.2012, letzte Änderung: 29.08.2024