Gigaframe.Com.PowerShell.CGI Web App Handler & Library

Table of Contents

Introduction

In today's fast paced ecosystem of numerous technology frameworks, and if you've resided in the technology industry long enough, gradually moving from either one area to another, one project to another, or role to another, you come to realize there is an increasing and exuberant need for simplicity to combat ever more demanding, complex and challenging times. For developers who move into systems management and/or systems administrative roles and/or Cybersecurity roles, you quickly find that your programming, analytical and solutioning skills and deep low-level understanding of all things technology are invaluable and highly transferable into numerous systems and infrastructure management initiatives and changes by employing the many solutions and design patterns known and unknown (ingenuity) are required to address the constantly changing world. For any technology person,

the need for programming skills is a requirement in a variety of industries as it provides a means to deliver efficiencies and exponentially handle workloads and deliver outcomes. Programming and its inherently associated skills, in today's world can make one man an army.
For most in this modern ecosystem of technology, architectures, systems, their management and solution design to support necessary or evolutionary changes, PowerShell has become an indispensable tool for many reasons, but primarily for its sheer broad powers and its simplicity. Microsoft has done a great job with PowerShell. Many developers will know that PowerShell is an extension or alternative scripted language for an underlining .NET framework, but it has much more, especially on Windows systems, which recently with .NET Core/5+ has become cross-platform.
Even though PowerShell can do many things, such as making web requests to web services your network or on the Internet, one thing that has been missing was the ability to develop web applications using PowerShell scripting. We hope to change that with this CGI processor and a set of supporting web application scripted libraries and demo apps.
With a common Client and Server scripting language and framework, you don't have employ the complexities that are inherent when incorporating a different server-side web app framework. And this also helps with sustainability, such as continuity of services in the life-cycle and change management and training of new staff, as PowerShell is widely used and appears will continue to be used deep into the future, especially now that it recently shifted to cross-platform, especially moving into the Linux word.
With this CGI handler and supporting scripts, you can develop your desktop management scripts as you do now and develop your server-side web services that can accept those web calls using the same scripting language and framework, harnessing the power of PowerShell. For example, no compiling. You can also develop web apps that encompass the whole spectrum of modern-day web app functionality (using HTML5 and modern JavaScript libraries of course) and also generate reports from the server or any other output using PowerShell as a web app framework.
What is Old and New Again!

Potential Uses

Given this was a fun little project with limited testing, the handler and scripts are intended for learning on small scale solutions on private networks. Further testing is required for it to be Internet ready.
In this scope:

- Learning web programming
- Building a custom web UI for IT administrative processes
- Building simple web services to receive computer communications
- Building online reports
- Building web apps

Install PowerShell

Install Powershell from "Turn Windows Features On" under "Control Panel" and "Programs and Features"

Install PowerShell 7.0+:
Installing PowerShell on Windows - PowerShell | Microsoft Learn
To install PowerShell on Windows, use the following links to download the install package from GitHub.

- PowerShell-7.2.6-win-x64.msi
- PowerShell-7.2.6-win-x86.msi

On Linux:
Install PowerShell on Linux - PowerShell | Microsoft Learn

Install Editor (Visual Code)

Perhaps the best code editor on the market, at least best free editor, is Microsoft's Visual Code. It is cross platform and offers many features.
Download Visual Studio Code - Mac, Linux, Windows

Install Database Server

There are minimally no limits to which databases you can use with Gigaframe.Com.PowerShell.CGI as that limit if restricted by PowerShell and underlining .NET itself, but there are 2 script libraries that come with the software you can use to connect to MSSQL and MariaDB (free variant of MySQL).

Install Microsoft SQL Server

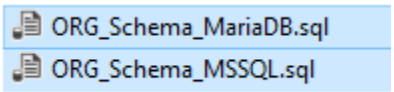
To Install MS SQL Server (Full, Express, or Developer), go this page, download and install the engine and SQL Management Tools
SQL Server Downloads | Microsoft

Install MariaDB

Recommend version 10, minimally version 10.5 version, and can be found here:
MariaDB Products & Tools Downloads | MariaDB
Install the version of your choice, but Community is a free option that works.
Also download needed connector for PowerShell and .NET based applications to use:
Download MariaDB Connectors for data access & analysis | MariaDB
All Files - C connector (mariadb.com)
All Files - ODBC connector (mariadb.com)
MySQL :: Download Connector/NET
Must install "mysql-connector-net-8.0.30.msi" minimally
Ensure to install HeidiSQL management tool.

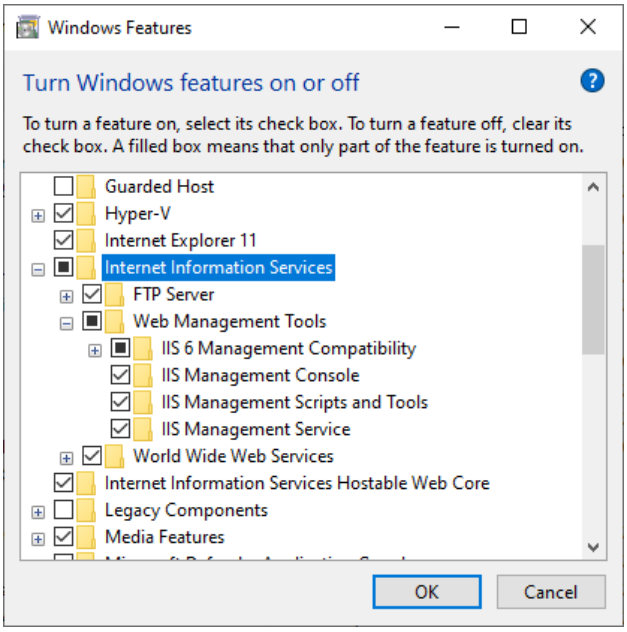
Install Sample Database in MSSQL and MariaDB

Under the SampesDatabase folder you'll find two SQL files for MSSQL and MariaDB respectively to produce the same database.
Use each in each respective database server.



Install Web App Server

There are two options for web application servers Gigaframe.Com.powerShell.CGI can be use and handled through: Windows own IIS Server, and Apache HTTP Server.
On Windows you can install IIS which is built for Windows by Microsoft and is free with most Windows versions. You need Windows Server to process web requests for many users. Other versions have limits but are good for development.
Install Internet Information Services (Web App Server on Windows) on Windows 10+ Pro:
Control panel-> Select Program-> Turn Windows features on and off
And Select "Internet Information Services" and install



Or install IIS Express:
Download Internet Information Services (IIS) 10.0 Express from Official Microsoft Download Center

Install Apache Web Server

On Windows:
Using Apache HTTP Server on Microsoft Windows - Apache HTTP Server Version 2.4
On Linux:
Download - The Apache HTTP Server Project

Setup IIS

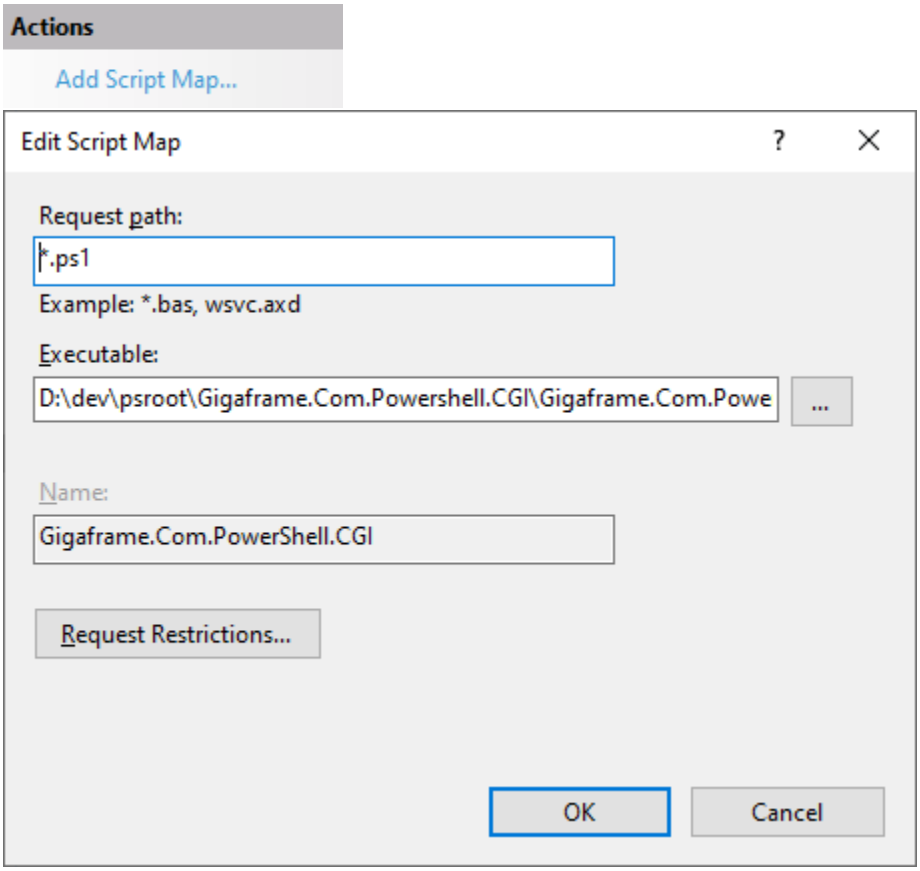
On IIS on Windows, add CGI and handler to process PowerShell scripts and render web output
Set Handler Script Mapped to:
D:\dev\psroot\Gigaframe.Com.Powershell.CGI\Release\Gigaframe.Com.PowerShell.CGI.exe %s %s

The CGI script handler is explained further below but same script handler can be used on Apache HTTP Server.

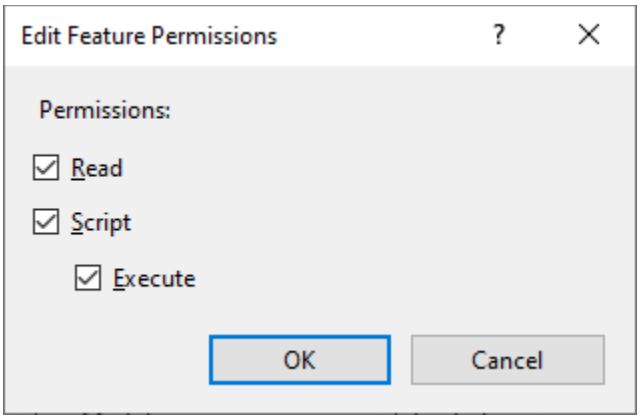
The PowerShell CGI Handler

The basis for this is a mapping is:
c:\windows\system32\WindowsPowerShell\v1.0\powershell.exe -NoLogo -NoProfile -NonInteractive -ExecutionPolicy Unrestricted -File %s %s

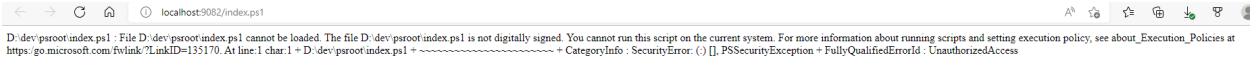
But it became very obvious this required something additional process CGI requests for PowerShell to handle. The CGI script handler address this gap.
So the IIS Script Hander mapping bust be assigned to this:



You may have to set the Feature Permission to allow Execute
This right-click the entry and select “Edit Feature Permission”, and check “Execute” and click OK.

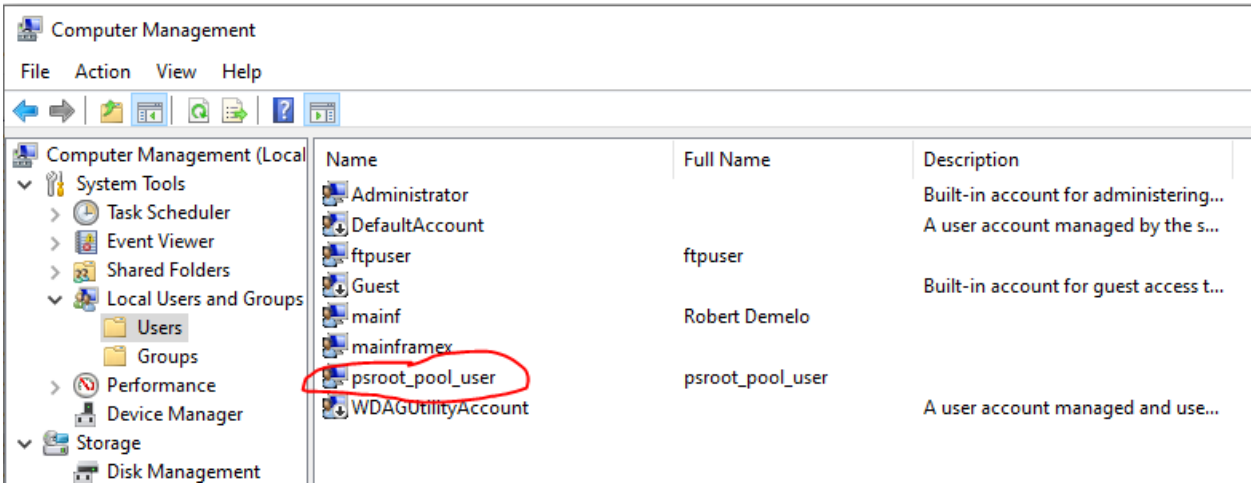


This worked but resulted in a the following error:

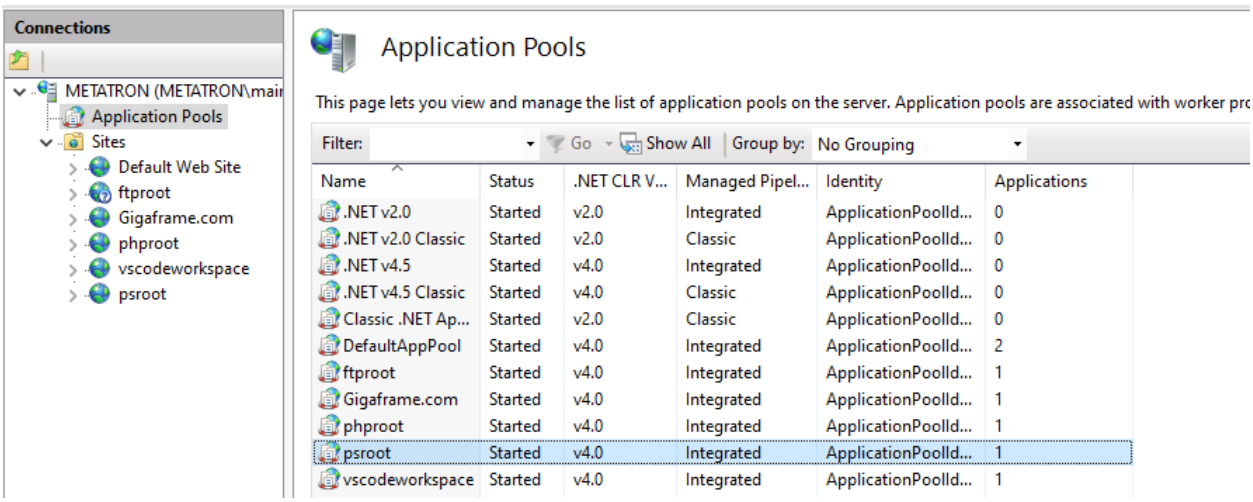


D:\dev\psroot\index.ps1 : File D:\dev\psroot\index.ps1 cannot be loaded. The file D:\dev\psroot\index.ps1 is not digitally signed. You cannot run this script on the current system. For more information about running scripts and setting execution policy, see about_Execution_Policies at https:/go.microsoft.com/fwlink/?LinkID=135170. At line:1 char:1 + D:\dev\psroot\index.ps1 + ~~~~~~~~~~~~~~~~~~~~~~~~ + CategoryInfo : SecurityError: (:) [], PSSecurityException + FullyQualifiedErrorId : UnauthorizedAccess

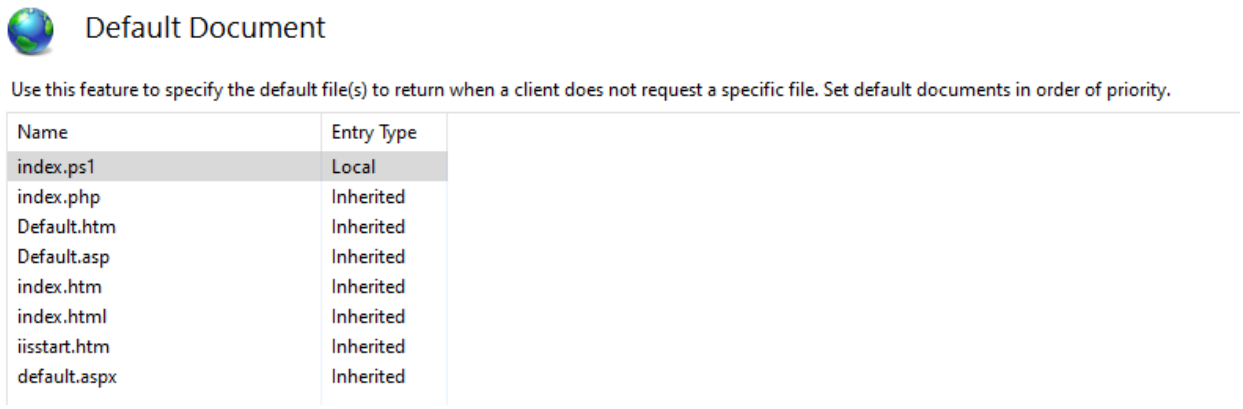
To resolve this create a local user as service account for the IIS application pool



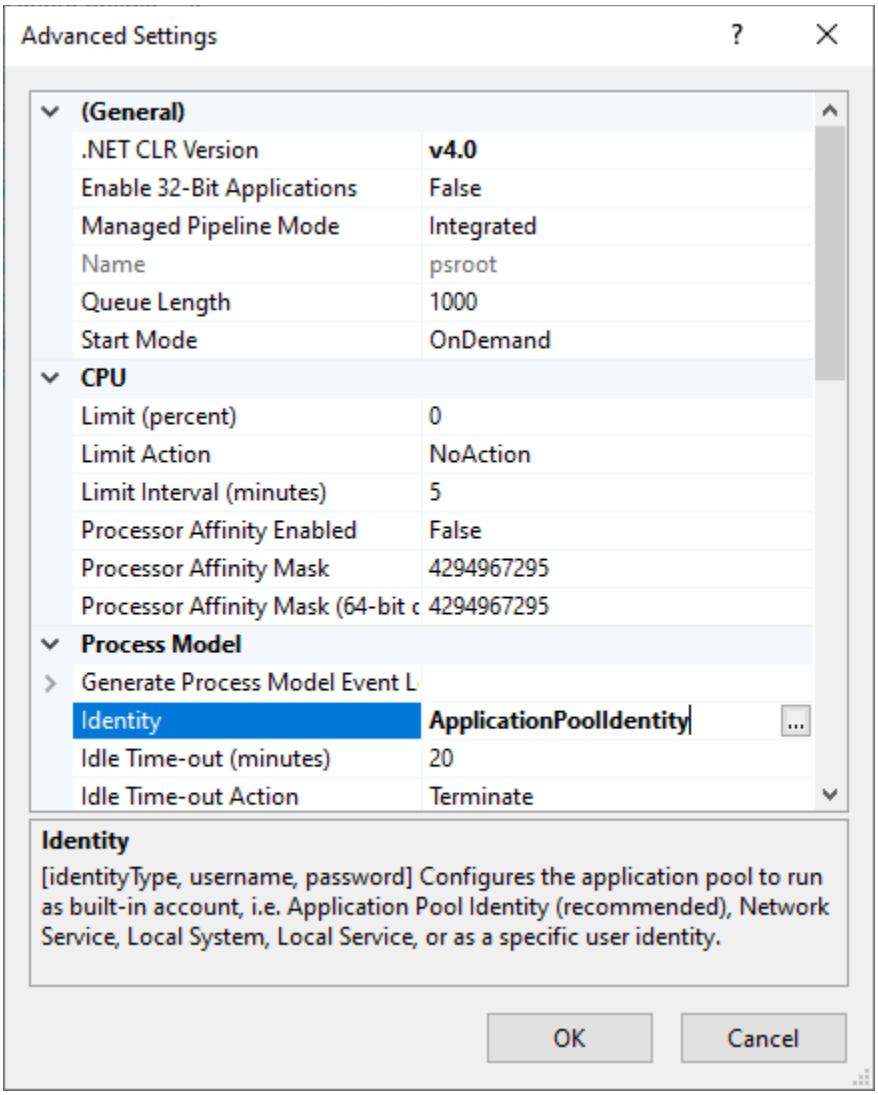
REMOVE on distributed release: password on test machine devpassword123!

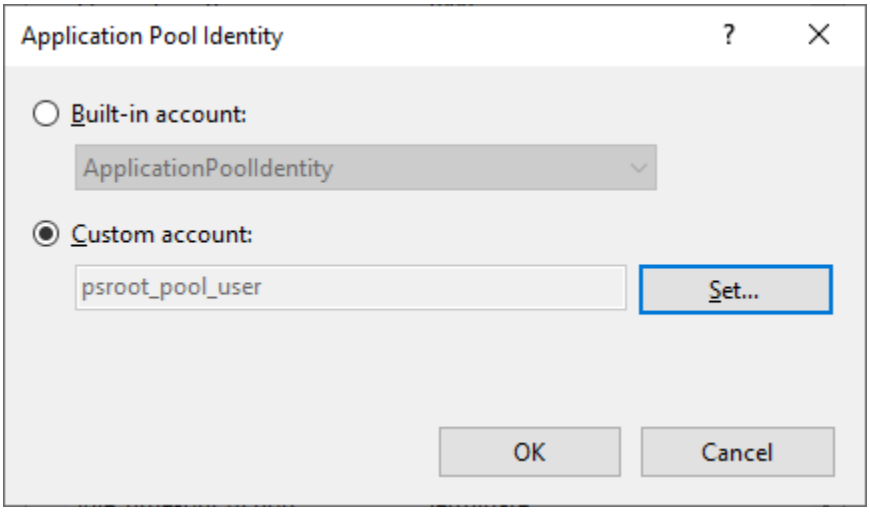


Add index.ps1 to default document:



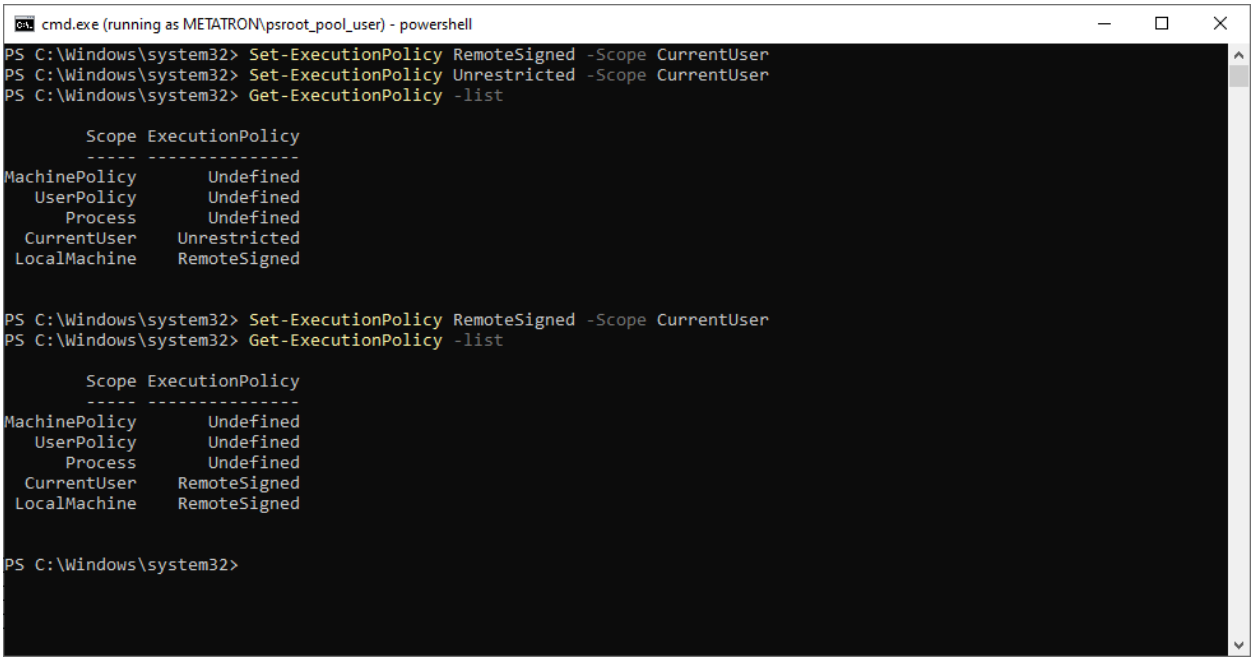
New create a Website or Application under your main Website and assign the created app pool.
The path of the Website or Application should the path to where you placed Gigaframe.Com.PowerShell.CGI/psroot folder.
Change the application pool user to the user you created.


Advanced Settings
(General)
.NET CLR Version
v4.0
Enable 32-Bit Applications
False
Managed Pipeline Mode
Integrated
Name
psroot
Queue Length
1000
Start Mode
OnDemand
CPU
Limit (percent)
0
Limit Action
NoAction
Limit Interval (minutes)
5
Processor Affinity Enabled
False
Processor Affinity Mask
4294967295
Processor Affinity Mask (64-bit c
4294967295
Process Model
Generate Process Model Event L
Identity
ApplicationPoolIdentity
Idle Time-out (minutes)
20
Idle Time-out Action
Terminate
Identity
[identityType, username, password] Configures the application pool to run as built-in account, i.e. Application Pool Identity (recommended), Network Service, Local System, Local Service, or as a specific user identity.
OK
Cancel


Application Pool Identity
Built-in account:
ApplicationPoolIdentity
Custom account:
psroot_pool_user
Set...
OK
Cancel

This logon through CMD with RUNAS to set the execution policy for this user on the Windows system.
Setting scope Current User to RemoteSigned. You can test with Unrestricted.
Restart IIS and Recycle your application pool.
>Powershell
>runas /user:”psroot_pool_user” cmd.exe
Enter the password of the user.

```
C:\Users\mainf>runas /user:"psroot_pool_user" cmd.exe
Enter the password for psroot_pool_user:
Attempting to start cmd.exe as user "METATRON\psroot_pool_user" ...

C:\Users\mainf>
```



Setup Apache

In the installation folder or tour Apache, spen httpd.conf apache configuration file.
Ensure this section in the file looks like this:

```
<Directory "${SRVROOT}/cgi-bin">
    Options +FollowSymLinks +ExecCGI
    AllowOverride None
    Require all granted
</Directory>
```

Then “AddHandler”, uncomment if needed, and add “.ps1” to the end of that line

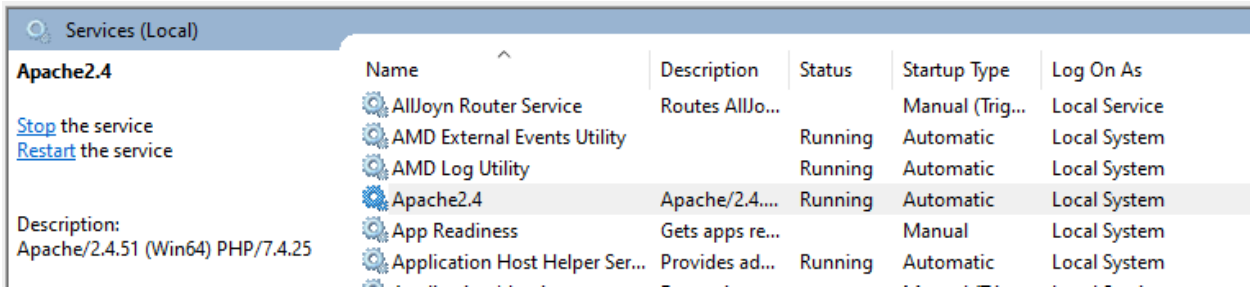
```
#
    # AddHandler allows you to map certain file extensions to "handlers":
    # actions unrelated to filetype. These can be either built into the server
```

```
    # or added with the Action directive (see below)
    #
    # To use CGI scripts outside of ScriptAliased directories:
    # (You will also need to add "ExecCGI" to the "Options" directive.)
    #
    AddHandler cgi-script .cgi .ps1
```

They add index.ps1 to this section, which informs Apache which file to look for on a directory call:

```
<IfModule dir_module>
    DirectoryIndex index.ps1 index.cgi index.html
</IfModule>
```

Save the file.
Restart the Apache server, either in Windows Services, or command line.

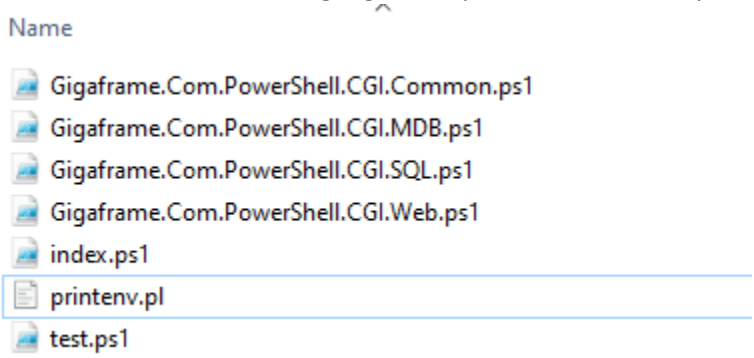


OR cmd:
> net stop Apache2.4
> net start Apache2.4
To start Apache 2.4 web server on Linux, enter:
/etc/init.d/apache2 start
OR sudo /etc/init.d/apache2 start
OR sudo service apache2 start.
Next develop your first Powershell .ps1 script, save it into your psroot folder mapped in IIS or the cgi-bin used by Apache server.
For Apache CGI, the first line tells Apache engine where to find the executable to handle the processing of the file.

```
#!D:\dev\psroot\Gigaframe.Com.Powershell.CGI\Gigaframe.Com.Powershell.CGI.exe
<#HEADER#>
Write-Host("Content-Type: text/html");
Write-Host("");
<#HTML#>
Write-Host("<html>");
Write-Host("<body>");
Write-Host("Hello World!");
Write-Host("</body>");
Write-Host("</html>");
```

Demo PowerShell Web App

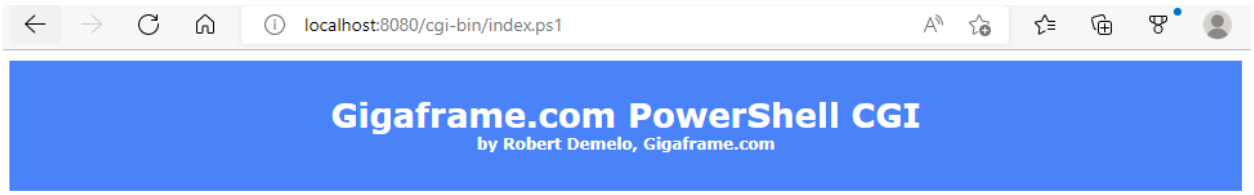
This software provides a demo PowerShell web app (index.ps1) that includes some very simple yet important script libraries to inline include with your web ps1 apps. This script libraries provide a set of methods and pre-processing to help develop PowerShell Web Apps like traditional web apps using different frameworks and languages. They are located in the psroot folder.



Inline script libraries included:

```
<# INLINE INCLUDES #>
(. $PSScriptRoot\Gigaframe.Com.PowerShell.CGI.Common.ps1);
(. $PSScriptRoot\Gigaframe.Com.PowerShell.CGI.Web.ps1);
(. $PSScriptRoot\Gigaframe.Com.PowerShell.CGI.SQL.ps1);
(. $PSScriptRoot\Gigaframe.Com.PowerShell.CGI.MDB.ps1);
```

The supplied psroot folder contain an index.ps1 file with functionality used in web applications, such as Server Variables, GET and POST handling functions, and database access to MSSQL and MDB



Hello World!

Server Vars
QUERY_STRING:
GATEWAY_INTERFACE: CGI/1.1
SERVER_NAME: localhost
SERVER_SOFTWARE: Apache/2.4.51 (Win64) PHP/7.4.25
SERVER_PROTOCOL: HTTP/1.1
SERVER_PORT: 8080
SERVER_PORT_SECURE:
REQUEST_METHOD: POST
PATH_INFO:
PATH_TRANSLATED:
REMOTE_ADDR: ::1
HTTP_CONTENT_LENGTH:
AUTH_TYPE:
CONTENT_TYPE: application/x-www-form-urlencoded
CONTENT_LENGTH: 46
LOGON_USER:

Click Here to Test Query String
Name Robert
Age 39
1st Number: 11
2nd Number: 11
Submit

Query Data

Posted Data
txtname=Robert
txtage=39
txtnum1=11
txtnum2=11
Total of 1st + 2nd Numbers = 22

Getting Data From MSSQL using Gigaframe.Com.PowerShell.SQL.SQL

UserID	FirstName	LastName	Email	Phone
RDEMELOP	Robert	Demelo	RDEMELOP@GIGAFRAME.COM	555-555-5555
RFRANKLIN	Randy	Franklin	RFRANKLIN@GIGAFRAME.COM	555-555-5555
RMERCURY	Rachel	Mercury	RMERCURY@GIGAFRAME.COM	555-555-5555

Getting Data From MariaDB using Gigaframe.Com.PowerShell.CGI.MDB

UserID	FirstName	LastName	Email	Phone
RDEMELOP	Robert	Demelo	RDEMELOP@GIGAFRAME.COM	555-555-5555
RFRANKLIN	Randy	Franklin	RFRANKLIN@GIGAFRAME.COM	555-555-5555
RMERCURY	Rachel	Mercury	RMERCURY@GIGAFRAME.COM	555-555-5555

```
#!D:\dev\psroot\Gigaframe.Com.Powershell.CGI\Gigaframe.Com.PowerShell.CGI.exe
Using Namespace System.Collections.Generic;
<#HEADER#>
#Write-Host("HTTP/1.1 200 OK");
Write-Host("Content-Type: text/html");
Write-Host("");
<# INLINE INCLUDES #>
(. $PSScriptRoot\Gigaframe.Com.PowerShell.CGI.Common.ps1);
(. $PSScriptRoot\Gigaframe.Com.PowerShell.CGI.Web.ps1);
(. $PSScriptRoot\Gigaframe.Com.PowerShell.CGI.SQL.ps1);
(. $PSScriptRoot\Gigaframe.Com.PowerShell.CGI.MDB.ps1);
<# BODY #>
Write-Host("<html>");
Write-Host("<head>");
Write-Host("<title>Gigaframe.com Powershell CGI Demo</title>");
```

```
Write-Host("</head>");
Write-Host("<body style=""font-family:arial;"">");
Write-Host("<div style=""width:100%;height:75px;background-
color:#4a82f7;color:#ffffff;text-align:center;vertical-align:middle;font-
family:verdana;font-size:26px;font-weight:bold;border-color:#0f3585;padding-
top:26px;"">");
Write-Host("Gigaframe.com PowerShell CGI");
Write-Host("<br/><span style=""font-size:12px;"">by Robert Demelo,
Gigaframe.com</span>");
Write-Host("</div>");
Write-Host("<br/><br/>");
Write-Host("<b>Hello World!</b><br/><br/>");

Write-Host("<table style=""width:100%"">");
Write-Host("<tr>");

Write-Host("<td>");
Write-Host("<b>Server Vars</b><br/>");
Write-Host("QUERY_STRING: " + $QueryData + "<br/>");
Write-Host("GATEWAY_INTERFACE: " +
[System.Environment]::GetEnvironmentVariable("GATEWAY_INTERFACE") + "<br/>");
Write-Host("SERVER_NAME: " +
[System.Environment]::GetEnvironmentVariable("SERVER_NAME") + "<br/>");
Write-Host("SERVER_SOFTWARE: " +
[System.Environment]::GetEnvironmentVariable("SERVER_SOFTWARE") + "<br/>");
Write-Host("SERVER_PROTOCOL: " +
[System.Environment]::GetEnvironmentVariable("SERVER_PROTOCOL") + "<br/>");
Write-Host("SERVER_PORT: " +
[System.Environment]::GetEnvironmentVariable("SERVER_PORT") + "<br/>");
Write-Host("SERVER_PORT_SECURE: " +
[System.Environment]::GetEnvironmentVariable("SERVER_PORT_SECURE") + "<br/>");
Write-Host("REQUEST_METHOD: " +
[System.Environment]::GetEnvironmentVariable("REQUEST_METHOD") + "<br/>");
Write-Host("PATH_INFO: " +
[System.Environment]::GetEnvironmentVariable("PATH_INFO") + "<br/>");
Write-Host("PATH_TRANSLATED: " +
[System.Environment]::GetEnvironmentVariable("PATH_TRANSLATED") + "<br/>");
Write-Host("REMOTE_ADDR: " +
[System.Environment]::GetEnvironmentVariable("REMOTE_ADDR") + "<br/>");
Write-Host("HTTP_CONTENT_LENGTH: " +
[System.Environment]::GetEnvironmentVariable("HTTP_CONTENT_LENGTH") + "<br/>");
Write-Host("AUTH_TYPE: " +
[System.Environment]::GetEnvironmentVariable("AUTH_TYPE") + "<br/>");
Write-Host("CONTENT_TYPE: " +
[System.Environment]::GetEnvironmentVariable("CONTENT_TYPE") + "<br/>");
```

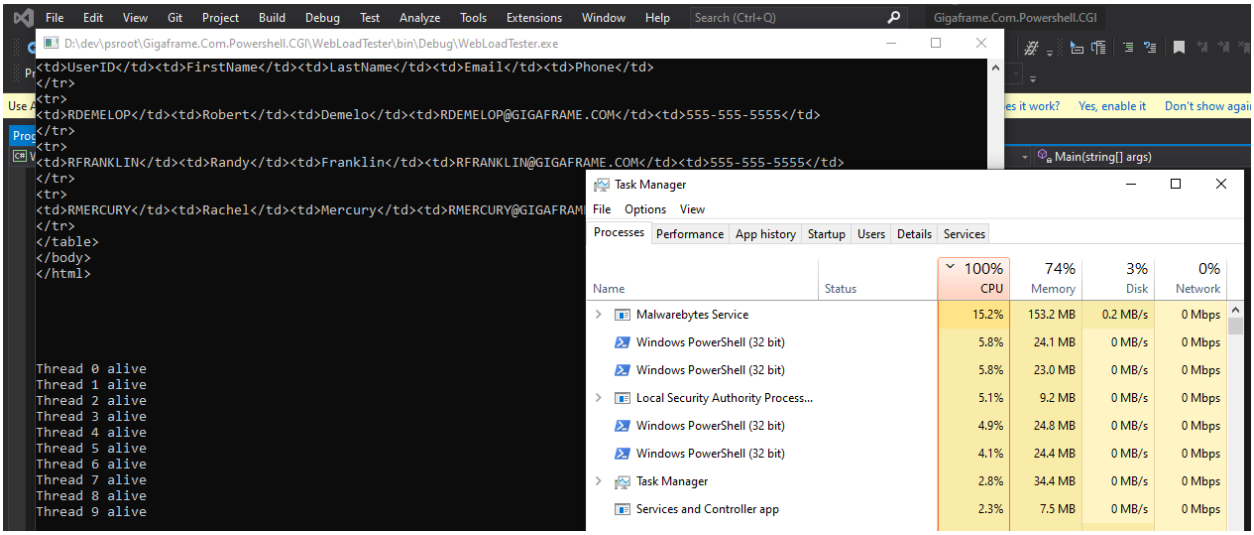
```
Write-Host("CONTENT_LENGTH: " +
[System.Environment]::GetEnvironmentVariable("CONTENT_LENGTH") + "<br/>");
Write-Host("LOGON_USER: " +
[System.Environment]::GetEnvironmentVariable("LOGON_USER") + "<br/>");
Write-Host("</td>");

Write-Host("<td>");
Write-Host("<a href=""?param1=Robert&param2=Developer"">Click Here to Test Query
String</a>");
Write-Host("<form action=""index.ps1"" method=""post"">");
Write-Host("Name <input type=""text"" id=""txtname"" name=""txtname"" value=""$
(GetPost("txtname"))""><br/>");
Write-Host("Age <input type=""text"" id=""txtage"" name=""txtage"" value=""$
(GetPost("txtage"))""><br/>");
Write-Host("1st Number: <input type=""text"" id=""txtnum1"" name=""txtnum1""
value=""$(GetPost("txtnum1"))""><br/>");
Write-Host("2nd Number: <input type=""text"" id=""txtnum2"" name=""txtnum2""
value=""$(GetPost("txtnum2"))""><br/>");
Write-Host("<input type=""submit"" id=""btnSubmit"" value=""Submit"" /><br/>");
Write-Host("</form>");
Write-Host("</td>");

Write-Host("<td>");
Write-Host("<b>Query Data</b><br/>");

# Parse Query String into Array
if($QueryData.Length -gt 0)
{
    ForEach($KeyName in $QueryVars.Keys)
    {
        Write-Host("$KeyName=$($QueryVars[$KeyName]) <br/>");
    }
}

Write-Host("<br/>");
Write-Host("<b>Posted Data</b><br/>");

if($PostData.Length -gt 0)
{
    ForEach($KeyName in $PostVars.Keys)
    {
        Write-Host("$KeyName=$($PostVars[$KeyName]) <br/>");
    }
}
```

```
$txtnum1 = GetPost("txtnum1");
$txtnum2 = GetPost("txtnum2");
If($txtnum1.Length -gt 0 -and $txtnum2.Length -gt 0)
{
    $num1 = [Convert]::ToInt32($txtnum1);
    $num2 = [Convert]::ToInt32($txtnum2);
    Write-Host("Total of 1st + 2nd Numbers = $($num1+$num2)");
}

Write-Host("</td>");

Write-Host("</tr>");
Write-Host("</table>");

<####### MSSQL Query + Results #######>

Write-Host("<h2>Getting Data From MSSQL using
Gigaframe.Com.PowerShell.SQL.SQL</h2>");

#[string] $ConnStr = "Server=(local);Database=Org;Trusted_Connection=True";
[string] $ConnStr = "Server=(local);Database=Org;User
ID=dev;Password=develop123!!";
[string] $SQLTxt = "SELECT UserID, FirstName, LastName, Email, Phone FROM USERS
WHERE FirstName LIKE @FirstName";
[List[KeyValuePair[[string],[object]]]] $ParamList =
[List[KeyValuePair[[string],[object]]]]::new();
$ParamList.Add([KeyValuePair[[string],[object]]]::new("@FirstName","R%"));

$Ds = MSSQLQuery -ConnString $ConnStr -SQLStatement $SQLTxt -Params $ParamList;

If($_MSSQL_Error -eq $True)
{
    Write-Host("<div>Error: $_MSSQL_Error</div>");
    Write-Host("<div>ErrorMessage: $_MSSQL_ErrorMessage</div>");
}

Write-Host("<table>");
Write-Host("<tr>");
Write-Host("<td>UserID</td><td>FirstName</td><td>LastName</td><td>Email</
td><td>Phone</td>");
Write-Host("</tr>");
ForEach($r in $Ds.Tables)
{
    Write-Host("<tr>");
```

```
    Write-Host("<td>$($r.UserID)</td><td>$($r.FirstName)</td><td>$
($r.LastName)</td><td>$($r.Email)</td><td>$($r.Phone)</td>");
    Write-Host("</tr>");
}
Write-Host("</table>");

<####### MariaDB Query + Results #######>

Write-Host("<h2>Getting Data From MariaDB using
Gigaframe.Com.PowerShell.CGI.MDB</h2>");

#[string] $ConnStr = "Server=(local);Database=Org;Trusted_Connection=True";
[string] $ConnStr = "Server=127.0.0.1;Database=Org;User
ID=root;Password=mariadb157!";
[string] $SQLTxt = "SELECT UserID, FirstName, LastName, Email, Phone FROM USERS
WHERE FirstName LIKE @FirstName";
[List[KeyValuePair[[string],[object]]]] $ParamList =
[List[KeyValuePair[[string],[object]]]]::new();
$ParamList.Add([KeyValuePair[[string],[object]]]::new("@FirstName","R%"));

$Ds = MDBQuery -ConnString $ConnStr -SQLStatement $SQLTxt -Params $ParamList;

If($_MDB_Error -eq $True)
{
    Write-Host("<div>Error: $_MDB_Error</div>");
    Write-Host("<div>ErrorMessage: $_MDB_ErrorMessage</div>");
}

Write-Host("<table>");
Write-Host("<tr>");
Write-Host("<td>UserID</td><td>FirstName</td><td>LastName</td><td>Email</
td><td>Phone</td>");
Write-Host("</tr>");
ForEach($r in $Ds.Tables)
{
    Write-Host("<tr>");
    Write-Host("<td>$($r.UserID)</td><td>$($r.FirstName)</td><td>$
($r.LastName)</td><td>$($r.Email)</td><td>$($r.Phone)</td>");
    Write-Host("</tr>");
}
Write-Host("</table>");

Write-Host("</body>");
Write-Host("</html>");
```

Multi-Threaded Load Testing

A load testing VS project is provided that you can customize to test.



In the Demo PS1 web page script there is a section to read POSTED data and insert into MSSQL demo ORG database

```
<######## MSSQL Add Data ########>
$SAVEPRODUCT = $PostVars["SAVEPRODUCT"];
Write-Host("<div>Save POSTED data: $SAVEPRODUCT</div>");
If($SAVEPRODUCT -eq "YES")
{
  $DEVICEPRODUCTID = $PostVars["DEVICEPRODUCTID"];
  $DEVICEID = $PostVars["DEVICEID"];
  $PRODUCTVERSION = $PostVars["PRODUCTVERSION"];
  [string] $ConnStr = "Server=(local);Database=Org;User
ID=dev;Password=develop123!";
  [string] $SQLTxt = "INSERT INTO DEVICEPRODUCTS
(DEVICEPRODUCTID,DEVICEID,PRODUCTNAME,PRODUCTVERSION,LASTUPDATE) VALUES
(@DEVICEPRODUCTID,@DEVICEID,@PRODUCTNAME,@PRODUCTVERSION,@LASTUPDATE)";
  [List[KeyValuePair[[string],[object]]]] $ParamList =
  [List[KeyValuePair[[string],[object]]]]::new();
  $ParamList.Add([KeyValuePair[[string],[object]]]::new("@DEVICEPRODUCTID",
$DEVICEPRODUCTID));
  $ParamList.Add([KeyValuePair[[string],[object]]]::new("@DEVICEID",$DEVICEID));
  $ParamList.Add([KeyValuePair[[string],[object]]]::new("@PRODUCTNAME",
$DEVICEPRODUCTID));
  $ParamList.Add([KeyValuePair[[string],[object]]]::new("@PRODUCTVERSION",
$PRODUCTVERSION ));
  $ParamList.Add([KeyValuePair[[string],[object]]]::new("@LASTUPDATE",
[DateTime]::Now));
  $RetVal = MSSQLExecute -ConnString $ConnStr -SQLStatement $SQLTxt -Params
$ParamList;
```

```
}
```

Software: Gigaframe.Com.PowerShell.CGI

WARRANTY AND LIABILITY DISCLAIMER

THIS SOFTWARE AND ANY SUPPLIED AND GENERATED CODE IS PROVIDED ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR, CONTRIBUTORS OR ANY ASSOCIATED PARTIES BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.