How do stop application pools from recycling in IIS 7.5?

I have configured the following settings:

ProcessModel -> Idle Time-out (minutes) = 0

Recycling -> Regular Time Intervals (minutes) = 0

Are these settings enought to stop an application pool from recycling?

Trong Pools có 2 thuộc tính mà ta cần quan tâm.

* Idle Time-out(Minutes) =20 (giá trị default) tức là khi website 20 phút mà không có bất kỳ request nào thì website sẽ ở trạng thái Idle.

Khi có request tiếp theo thì website sẽ tự động dậy và vào application start nên quá trình này sẽ bị mất thời gian để tải trang.

Để giải quết vấn đề này ta sẽ để giá trị này trùng với giá trị Regular Time Interval.

* Regular Time Interval (minutes) default =1740 tương đương với 29h

Tức là với bất kỳ website nào thì khi khởi tạo nếu không chọn vào pool mặc định thì nó sẽ tự động sinh ra 1 pool có tên tương ứng với site đã tạo.

Thời gian 29h windows mặc định sẽ reset lại pools 1 lần quá trình này tương đương với quá trình khởi động nhanh dọn dẹp rác trên ứng dụng của windows.

Trong quá trình Recycling pool sẽ reset lại nhưng không tự động start appcation trong ứng dụng mà chỉ khi nào khi có request đầu tiên tới ứng dụng thì ApplicationStart mới bắt đầu chạy quá trình này sẽ mất một thời gian để hệ thống load dữ liệu trên memory. Nên dẫn tới request đầu tiên sẽ rất chậm vì đợi hệ thống khởi động và load dữ liệu trên mem.

Để giải quết vấn đề này ta sẽ tiến hành cấu hình trong InetService với file ApplicationHost

Bước 1. Vào thư mục C:\Windows\System32\inetsrv\config\applicationHost.config

Tìm tới thẻ applicationPools

Add thêm thông số startMode=”AlwaysRunning”

< applicationPools >

name="vsdportal" autoStart="true" enable32BitAppOnWin64="true" managedRuntimeVersion="v4.0" startMode="AlwaysRunning" />

</applicationPools>

Bước 2. Tiếp theo tìm tới thẻ sites add thêm thuộc tính serviceAutoStartEnabled="true"*vào thẻ* <application>

<sites>

<site name="vsdportal" id="1">

<application path="/" applicationPool="vsdportal" serviceAutoStartEnabled="true" serviceAutoStartProvider="PreWarmMyCache">

<virtualDirectory path="/" physicalPath="C:\inetpub\wwwroot\VSDPortal"/>

</application>

<bindings>

<binding protocol="http" bindingInformation="\*:80:" />

</bindings>

</site>

</sites>

Auto-Start ASP.NET Applications (VS 2010 and .NET 4.0 Series)

Tuesday, September 15, 2009

[**.NET**](http://weblogs.asp.net/scottgu/Tags/.NET) [**ASP.NET**](http://weblogs.asp.net/scottgu/Tags/ASP.NET) [**Community News**](http://weblogs.asp.net/scottgu/Tags/Community%20News) [**IIS7**](http://weblogs.asp.net/scottgu/Tags/IIS7) [**Visual Studio**](http://weblogs.asp.net/scottgu/Tags/Visual%20Studio)

This is the seventh in [a series of blog posts](http://weblogs.asp.net/scottgu/archive/2009/08/25/vs-2010-and-net-4-series.aspx) I’m doing on the upcoming VS 2010 and .NET 4 release.

I’m going to switch from discussing new VS 2010 tooling features and instead do a few posts covering a few new runtime features (don’t worry – I’ll come back to a lot more VS features, I’m just trying to mix things up a bit).

Today’s post covers a small, but nice, new feature that you can now optionally take advantage of with ASP.NET 4 - the ability to automatically startup and proactively initialize a web application without having to wait for an external client to hit the web server.  This can help you provide a faster response experience for the first user who hits the server, and avoids you having to write custom scripts to “warm up” the server and get any data caches ready.  It works with all types of ASP.NET applications – including both ASP.NET Web Forms and ASP.NET MVC based applications.

Auto-Start Web Applications with ASP.NET 4

Some web applications need to load large amounts of data, or perform expensive initialization processing, before they are ready to process requests.  Developers using ASP.NET today often do this work using the “Application\_Start” event handler within the Global.asax file of an application (which fires the first time a request executes).  They then either devise custom scripts to send fake requests to the application to periodically “wake it up” and execute this code before a customer hits it, or simply cause the unfortunate first customer that accesses the application to wait while this logic finishes before processing the request (which can lead to a long delay for them).

ASP.NET 4 ships with a new feature called “auto-start” that better addresses this scenario, and is available when ASP.NET 4 runs on IIS 7.5 (which ships with Windows 7 and Windows Server 2008 R2).  The auto-start feature provides a controlled approach for starting up an application worker process, initializing an ASP.NET application, and then accepting HTTP requests.

Configuring an ASP.NET 4 Application to Auto-Start

To use the ASP.NET 4 auto-start feature, you first configure the IIS “application pool” worker process that the application runs within to automatically startup when the web-server first loads. You can do this by opening up the IIS 7.5 applicationHost.config file (C:\Windows\System32\inetsrv\config\applicationHost.config) and by adding a*startMode=”AlwaysRunning”* attribute to the appropriate <applicationPools> entry:

<applicationPools>

     <add name="MyAppWorkerProcess" managedRuntimeVersion="v4.0"**startMode="AlwaysRunning"** />

</applicationPools>

If you load up the Windows task manager, click the “show processes from all users” checkbox, and then hit save on a startMode attribute change to the applicationHost.config file, you’ll see a new “w3wp.exe” worker process immediately startup as soon as the file is saved.

A single IIS application pool worker process can host multiple ASP.NET applications.  You can specify which applications you want to have automatically start when the worker process loads by adding a *serviceAutoStartEnabled="true"*attribute on their <application> configuration entry:

Luôn để ID =1

<site name="HNX4TAdmin" id="1" serverAutoStart="true">

<sites>

     <site name="MySite" id="1">

          <application path="/" **serviceAutoStartEnabled="true"serviceAutoStartProvider="PreWarmMyCache"**/>

     </site>

</sites>

<serviceAutoStartProviders>

     <add name="PreWarmMyCache" type="Administrator.PreWarmCache, Administrator" />

</serviceAutoStartProviders>

The *serviceAutoProvider="PreWarmMyCache"*attribute above references a provider entry within the config file that enables you to configure a custom class that can be used to encapsulate any "warming up" logic for the application.  This class will be automatically invoked as soon as the worker process and application are preloaded (before any external web requests are received), and can be used to execute any initialization or cache loading logic you want to run before requests are received and processed:

public class PreWarmCache : System.Web.Hosting.IProcessHostPreloadClient {

    public void Preload(string[] parameters) {

        // Perform initialization and cache loading logic here...

    }

}

IIS will start the application in a state during which it will not accept requests until your "warming up" logic has completed.  After your initialization code runs in the Preload method and the method returns, the ASP.NET application will be marked as ready to process requests.

You can optionally combine the new auto-start "warming up" feature with the load-balancing capabilities of the [IIS7 Application Request Routing (ARR)](http://www.iis.net/extensions/ApplicationRequestRouting) extension, and use it to signal to a load-balancer once the application is initialized and ready to accept HTTP traffic – at which point the server can be brought into the web farm to process requests.

Summary

The new "auto start" feature of ASP.NET 4 and IIS 7.5 provides a well-defined approach that allows you to perform expensive application startup and pre-cache logic that can run before any end-users hit your application.  This enables you to have your application "warmed up" and ready from the very beginning, and deliver a consistent high performance experience.

Hope this helps,

Scott

P.S. In addition to blogging, I have been using Twitter more recently to-do quick posts and share links.  You can follow me on Twitter at: <http://www.twitter.com/scottgu> (@scottgu is my twitter name)