PHP 5.3 auf Windows

Anleitung zum Glücklichsein

Mit PHP 5.3 und FastCGI wird der IIS ein ernstzunehmender Kandidat für den PHP-Produktivbetrieb auf Webservern. Der schnellste Weg zur Verwendung führt über den Web Platform Installer. Von Jan Schenk

Info

Auf einen Blick

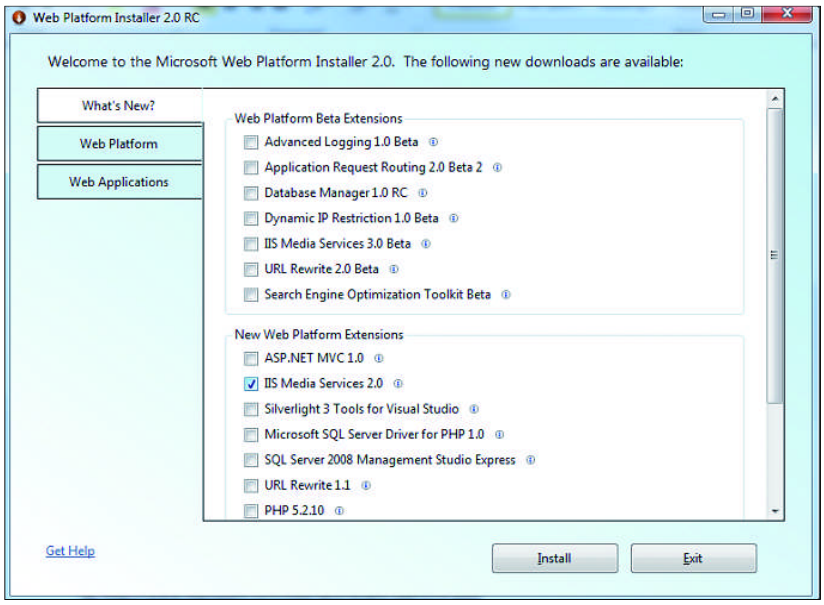
»Mit der neuen Version 5.3 von PHP wird auch im Windows-Bereich einiges anders.

»Dank des Engagements sowohl seitens Microsoft als auch des Core-Development-Teams von PHP wird die Konstellation Windows-IIS-PHP zu einer ernsthaften Alternative zu Linux-Apache-PHP..

»Ok, zum Entwickeln – so oder so ähnlich würden die meisten Web-Developer die Kombination PHP und Windows beschreiben. Und lange haben sie damit nicht ganz unrecht gehabt. Lange genug. Denn mit Version 5.3 und dem eng verknüpftem Engagement sowohl seitens Microsoft als auch des Core-Development-Teams wird nun vieles anders. Wie diese Neuigkeiten in der Community aufgenommen werden, kann allerdings letztlich nur die Zeit zeigen.

PHP auf Windows ist ja an sich nichts Neues. In den verschiedensten Variationen: WAMP, WIMP (mit IIS), WISP (mit SQL-Server), sogar WIOP oder WAOP (mit Oracle-Datenbank). Auf Entwicklermaschinen finden sich häufig erstere beiden. In der freien Wildbahn, also auf den Webservern im World Wide Web, läuft der Löwenanteil der PHP-Server im LAMP-Stack. Linux, Apache, MySQL und PHP. Was viele nicht wissen: Die Stack-Alternative auf Windows, der sogenannte WISP-Stack (steht für Windows, IIS, SQL-Server und PHP), hat einige Veränderungen durchgemacht. Und bietet seither viele Vorteile, aber kaum Nachteile für den Einsatz mit PHP. Im Folgenden soll die Entwicklung der Reihe nach erläutert werden.

Warum der von Microsoft kostenlos angebotene Internet Information Service (IIS) bei PHP-Deployments oft gar nicht zum Zuge kommt, obgleich er in der aktuellen Version 7 einiges zu bieten hat, ist klar: Seine Vorgänger haben sich bei der Pflege der PHP-Unterstützung nicht eben mit Ruhm bekleckert. Genau das soll nun anders werden. Neben der stetigen Entwicklung von Erweiterungen durch das IIS-Team und die Community (www.iis.net/extensions) unterstützt Microsoft seit einiger Zeit das Kernentwicklerteam von PHP. Warum nur, mag der ein oder andere Leser nun zu Recht zweifeln. Microsoft hat doch ein Konkurrenzprodukt zu PHP namens ASP.NET auf dem Markt. Die einfache Antwort? Microsoft ist sich bewusst geworden, dass es durchaus Platz und Daseinsberechtigung für beide Technologien gibt. Und dass es besser ist, beide Welten zu unterstützen, und am Ende glückliche und fachkompetente Entwickler zu haben, als zwei gegeneinander agierende Welten. Und wer behauptet, dass in einem Technologie-Stack nur Platz für eine der beiden Web-Sprachen (PHP oder ASP.NET) sein kann, der irrt sich. Der Internet Information Service macht vor, was Interoperabilität bedeutet. Und unterstützt PHP so gut wie nie zuvor.



Die Schritte zum und im Web Platform Installer sind denkbar einfach.

»Das FastCGI-Modul«

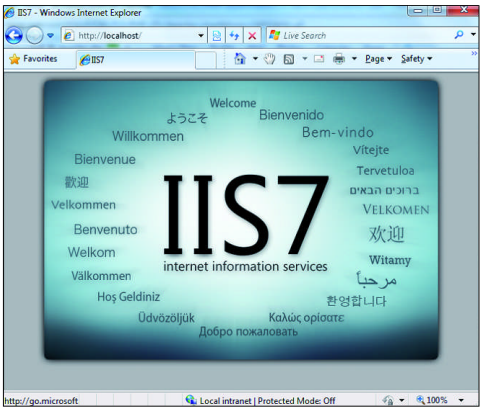
Tatsächlich bietet der IIS eine außergewöhnlich performante PHP-Unterstützung über das FastCGI-Modul, das in Zusammenarbeit mit der Firma Zend entstand. Hierbei wird den PHP-Calls eine Thread-sichere Umgebung zur Verfügung gestellt, bei der nicht jeder Webseiten-Aufruf einen einzelnen Thread erzeugt – auf Windows Betriebssystemen eine kostspielige Prozedur, sondern die Calls im FastCGI-Modul abgearbeitet werden. Dadurch entsteht ein gravierender Geschwindigkeitsvorteil gegenüber der Implementierung über die ISAPI-Schnittstelle. Ein weiterer Vorteil ist, dass die non-thread-safe-Variante von PHP verwendet werden kann, die keinen Overhead für eine eigene, interne Thread-Sicherheit erzeugt. Bei Verwendung der aktuellen PHP-Version 5.3 kann der IIS sowohl in Sachen Stabilität als auch in Fragen der Geschwindigkeit Punkte sammeln, sowohl gegenüber vorangegangenen Versionen, als auch gegenüber den alteingesessenen Mitbewerbern. Das allein sollte in jedem Fall ein Grund sein, den IIS ernsthaft in Erwägung zu ziehen, wenn es um die Plattformentscheidung geht.

Gleichzeitig kann man sich auch noch das integrierte Caching und die Media Services ansehen, die den ein oder anderen durchaus begeistern könnten. Außerdem ist der Internet Information Service 7 über den Microsoft Web Platform Installer (WebPI oder auch WPI) auf allen gängigen Windows-Installationen nachinstallierbar (www.microsoft.com/web/downloads/platform.aspx). Die Crux an der Sache ist, dass in den Web-Platform-Installer zur Fertigstellung dieses Artikels das neue PHP 5.3 noch nicht integriert war. Dort befindet sich derzeit noch Version 5.2.10. Wem das genügt, der ist mit dem WebPI auf dem einfachsten und schnellsten Weg unterwegs, um in weniger als 20 Minuten eine laufende PHP-Umgebung auf seinem Server respektive Desktop-System ans Laufen zu bekommen.

Die Schritte zum und im Web-Platform-Installer sind denkbar einfach: von der Seite www.microsoft.com/web/downloads/platform.aspx die Installationsdatei *wpilaun-*

cher.exe herunter laden, ausführen und eine beliebige Anwendung auswählen. Alle Abhängigkeiten werden vom WebPI soweit möglich automatisch aufgelöst: Bis auf MySQL funktioniert das auch hervorragend, den relationalen Schweden muss man von Hand installieren. Auch dafür gibt es unter der Adresse learn.iis.net/page.aspx/353/installing-and-configuring-mysql-for-php-applications-on-iis/eine Anleitung.

Der WebPI ist keine Software-Sammlung. Vielmehr stellt er ein Installationstool dar, welches Mechaniken integriert, um Software von den verschiedensten Quellen im Internet abhängigkeitenkorrekt und zentral zu installieren. Damit ist der WebPI selbst aber nicht in der Lage die Verfügbarkeit der einzelnen Komponenten zu gewährleisten, sondern muss sich auf die Mithilfe der Benutzer verlassen. So werden tote Links der Pakete immer



Gegebenenfalls muss der IIS in den Windows-Features aktiviert werden.

zuerst vom Anwender des WebPI entdeckt und sollten im Entwickler-für-Entwickler-Sinne an das WebPI-Team und den Bereitsteller der betroffenen Software gemeldet werden. Genug geredet, der IIS wartet darauf, sich mit PHP 5.3 beweisen zu dürfen. Falls wir inzwischen den WebPI und bereits das erste Paket installiert haben, ist der lokale IIS7 aktiviert worden. Für Windows-Varianten, die den IIS7 nicht von selbst mitbringen (alles von Windows XP SP2 und unter Windows 7 RC beziehungsweise Windows 2008 Server) empfiehlt sich dieser Weg über den WebPI ohnehin. Wir testen, ob der IIS läuft, indem wir die URL *http://localhost* aufrufen. Dort sollte uns die Begrüßungsgrafik des IIS7 erwarten.

»IIS in Windows aktivieren«

Falls nicht, ist es Zeit, den IIS in den Windows-Features zu aktivieren. Die Windows-Features erreicht man über die *Systemsteuerung, Programme hinzufügen/entfernen und Windows Features hinzufügen/entfernen*, oder über sie Suche im Startmenü und *Windows Features*. Wichtig ist der Punkt CGI, denn der sorgt dafür, dass wir PHP über das FastCGI-Modul andocken können.

Wer den IIS noch nie konfiguriert hat und bisher händisch *httpd.conf*-Dateien editiert hat, dem mag die Bedienung des grafischen Interfaces des IIS recht schwer fallen. Bunt, Mausbedienung und Kontextmenüs waren auch mir erst einmal fremd, wenn es um Webserver ging. Nach einiger Zeit gewöhnt man sich daran und lernt es

Autor

Jan Schenk ist Developer Evangelist bei Microsoft Deutschland. Dort ist er für die Bereiche Web und Live Services zuständig. Die Themen ASP.NET, Windows Azure, Silverlight und PHP Interoperability bilden seine Fokuspunkte. Jan Schenk hat acht Jahre Erfahrung im FreeSoftware-Umfeld gesammelt, bevor er 2008 zu Microsoft Deutschland wechselte. Er kennt die Vor- und Nachteile jeder Seite und spricht darüber gerne offen.

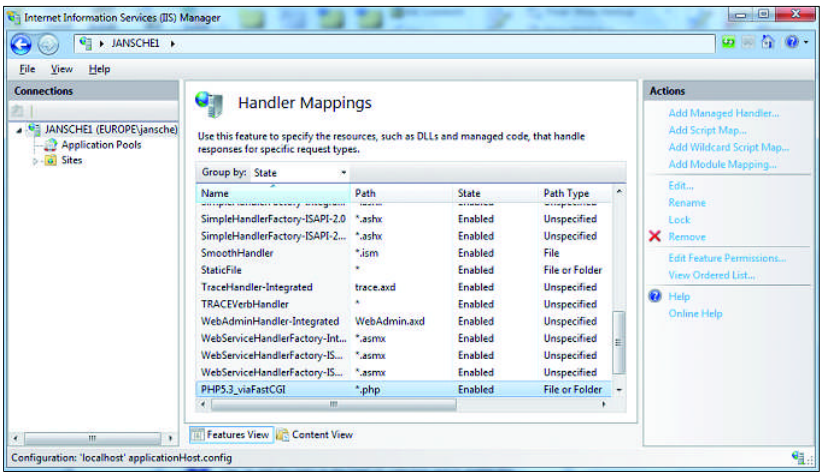
E blogs.msdn.com/jansche/

zu schätzen, dass man über einen einzigen Klick den gesamten Webserver-Dienst neu starten kann. Oder genauso einfach einen neuen *vhost* anlegen kann. Rechtsklick auf *Sites* und *Add Website* und schon bekommt man ein Dialogfenster mit den notwendigen Eingaben: Für jetzt begnügen wir uns aber mit den Standardeinstellugen in der *Default Web Site*. Die PHP-Runtime laden wir von der Projektseite windows.php.net/download/ herunter, und damit gleich zum ersten gut dokumentierten Stolperstein: Der seitlich erwähnte Hinweis über VC6 und VC9 lässt sich in der Vorfreude auf PHP 5.3 leicht übersehen. VC9 steht für Visual C++ 9, und bedeutet, dass die Quellen dieser Binärdateien mit der Visual Studio 2008-Entwicklungsumgebung kompiliert worden sind. Der neuere Compiler bringt gegenüber VC6 - dem inzwischen über zehn Jahre alten Compiler, mit dem alle bisherigen PHP-Binaries kompiliert worden sind - mehr Stabilität und Geschwindigkeit. Und das spürt man.

Die Installation der VC9-Variante setzt aber auch voraus, dass eine entsprechende Laufzeitumgebung auf der Zielmaschine vorhanden ist. Das notwendige Redistributionspaket findet sich im Microsoft-Download-Center unter www.microsoft.com/downloads/. Einmal heruntergeladen installiert man das drei MByte kleine Paket und ist damit für kommende PHP-Releases sorgenfrei.

Die Wahl beim PHP-Paket fällt bei uns auf die Zip-Version, die wir in das Verzeichnis *PHP5.3* unter *Programme* oder – für 64-Bit-Systeme - *Programme (x86)* nachinstallieren. Das Standardverzeichnis *PHP* im Programmordner wurde schon vom Web-Platform-Installer mit der Vorgängerversion von PHP belegt.

Ein Wort für Sympathisanten der *.msi*-Installer von der php.net-Seite: der Installer lässt nur eine lokale Installation von PHP 5.x zu. Werden die 5.3-Binaries über den Installer ins System geholt, so wird automatisch die



Hier wird für die PHP 5.3-Installation ein neuer Handler hinzugefügt.

Info

Apache vs. IIS – Wer ist schneller?

Mit Geschwindigkeitsvergleichen zwischen zwei Technologien tut man sich selten einen Gefallen.

Entweder rutscht man als Tester automatisch in eine voreingenommene Rolle, oder die Tests decken nur einen Bruchteil dessen ab, was für die unterschiedlichsten Applikationen relevant ist. Nichtsdestotrotz fordern Anwender und Entwickler immer wieder genau diesen schwierigen Vergleich. Einen Ansatz unter möglichst neutralen Bedingungen hat kürzlich Joe Stagner, Senior Program Manager bei Microsoft, auf seinem privaten Blog gewagt. Unter *http://misfitgeek.com/blog/aspnet/php-versus-asp-net-ndash-windows-versus-linux-ndash-who-rsquo-s-the-fastest/* lässt sich gut aufbereitet und anhand eigener Tests belegt nachlesen, dass es eben keinen generell schnelleren Webserver gibt, sondern die Einzelanforderungen entscheiden, welche Technologiekonstellation auch das letzte Quäntchen Geschwindigkeitsvorteil verspricht. Die Entscheidung wird mit diesem Wissen sicher nicht einfacher. Wer die Wahl hat, …

5.2.x-Umgebung unabhängig vom Installationsort entfernt. Dieser Umstand ist vermutlich der Integration in den IIS geschuldet, wenngleich der IIS mehrere verschiedene FastCGI-Applikationen – auch zwei verschiedene Versionen von PHP – zulässt, wie wir mit unserer Installation über die Zip-Datei gleich sehen werden. Möchte man unbedingt das *.msi*-Paket installieren, dann verzichtet man derzeit auf die Möglichkeit einer Parallelinstallation zweier PHP-Versionen. Das ist in den meisten Fällen kein großer Verzicht, besser aber man kennt die Haken und Ösen einer Multi-Version-Installation, die im Nachfolgenden noch weiter beschrieben werden.

Um nun den IIS auch noch dazu zu bringen, PHP 5.3 zu verwenden, rufen wir den IIS-Manager über die Startmenü-Suche und *inetmgr* oder *IIS Manager* auf. Dort fügen wir für unsere PHP 5.3 Installation einen neuen Handler hinzu. Nun legen wir zwei neue Applikationen in der Standard-Site des IIS an, *php5.2.10* und *php5.3*. Das geschieht, im Kontrast zum fehleranfälligeren manuellen Editieren einer Konfig-Datei , über einen Rechtsklick auf die *Default Web Site* und *Add Application*….

In der Anwendung *php5.3*, die schlüssiger Weise mit PHP 5.3 interpretiert werden soll, entfernen wir nun das gedoppelte Handler-Mapping wieder: Gleiches tun wir mit dem 5.3er-Handler-Mapping bei Anwendung *php5.2.10*, die mit PHP 5.2.10 geparst werden sollen. In diese beiden Verzeichnisse sollte mindestens noch eine *index.php*-Datei liegen, die zu Testzwecken eine *phpinfo()*-Funktion beherbergt.

Die Konfiguration des FastCGI-Moduls im IIS können wir vorerst getrost der Automatik überlassen, die wir nach dem Hinzufügen eines neuen Handler-Mappings erfahren. Wer hier noch an der Performance-Schraube drehen möchte, kann unter der Adresse learn.iis.net/page.aspx/ 246/using-fastcgi-to-host-php-applications-on-iis-70/ die passenden Tipps finden.

Die PowerShell erlaubt ein komfortables und effektives Arbeiten.

»PHP-Konfigurationsdatei«

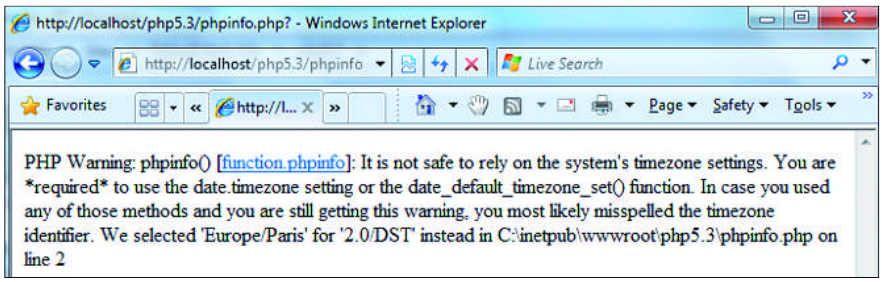
Es gibt aber leider einen weiteren Problempunkt beim Kombinieren von Installer- und Zip-basierten Installationen. So sollte man vermehrt einen Blick auf die von der PHP-Laufzeitumgebung eingelesene *php.ini* haben. Der Wert, von wo und welche *php.ini* eingebunden wird, befindet sich bei Installation über das *.msi* in der Registry. Um zu überprüfen, ob die richtigen *php.ini*-Dateien von der jeweiligen Installation verwendet werden, benutzen wir die simple Codezeile *<?php phpinfo(); ?>* in einer PHP-Seite unserer Wahl in der jeweiligen Applikation.

In der Ausgabe bekommen wir in der neunten Tabellenzeile mitgeteilt, woher die *php.ini* bezogen wird. Und siehe da, bei mir wird doch tatsächlich die *php.ini* der falschen Installation geladen: Anstelle des *C:\Program Files (x86)\PHP\php.ini* hätte ich hier *C:\Program Files (x86)\PHP5.3\php.ini* erwartet. Abhilfe schafft derzeit leider nur der händische Eingriff in die Registry über Start und *regedit*. Nach dem Löschen all der Werte, die *php.ini* im Datenfeld haben, fällt PHP in das Standard-Verhalten zurück, die *php.ini* von der gleichen Lokation der *php-cgi.exe* zu lesen. Und nur genau dieses Verhalten macht bei unserem Szenario Sinn.

Falls es mit dem Multiversionsszenario nicht sofort klappen sollte, und ein Aufruf der auf PHP5.3-gemappten Applikation eine Warnmeldung ausgibt, muss man nicht verzweifeln: Den fehlenden Wert für *date.timezone* in der richtigen *php.ini*-Datei ergänzen (Europe/Paris oder doch lieber Europe/Berlin), den Prozess *php-cgi.exe* im Task-Manager beenden oder den ApplicationPool über den IIS-Manager recyclen, und schon läuft die 5.3-Installation reibungslos. Nun da es mit der Grundinstallation rund läuft zum Feintuning. Windows-spezifische php.ini Werte, die einen Blick wert sind, gibt es nur eine gute Hand voll. Zu finden sind diese im folgenden Listing:

```
fastcgi.impersonate = 1
cgi.fix_pathinfo = 1
cgi.force_redirect = 0
extension_dir
allow_url_fopen = Off
allow_url_include = Off
register_globals = Off
open_basedir = "C:\inetpub\"
fastcgi.logging = 0
```

Weitere wertvolle Tipps zur php.ini finden sich unter learn.iis.net/page.aspx/246/using-fastcgi-to-host-php-applications-on-iis-70/#PHP_Security_Recommendations und auf Bernhard Franks TechNet-Blog unter blogs.technet.com/bernhard_frank/archive/2009/06/09/php-ini-einstellungen-f-r-php-auf-windows.aspx. Änderungen an der php.ini



Möglicherweise klappt ein Multiversionsszenario mit PHP 5.2 und 5.3 nicht auf Anhieb.

werden beim IIS durch ein Neuladen des ApplicationPools, dem die Applikation angehört, propagiert. Das lässt sich über den IIS Manager über die Ansicht der Pools ausführen, oder aber, für Freunde der Kommandozeile:

```
%WINDIR%\system32\inetsrv\appcmd
recycle APPPOOL DefaultAppPool
```

DefaultAppPool wird natürlich fallspezifisch mit dem Namen des verwendeten ApplicationPools ersetzt. Dafür bedarf es Administratorprivilegien auf der Shell. Ähnlich effektiv, wenn auch rabiater, ist das Beenden des php-cgi.exe-Prozesses über den Task Manager. Eleganter ist da sicherlich der Weg über die Shell. So lässt sich jeder Befehl anstatt in der IIS7 Manager UI auch auf der Kommandozeile mit dem Kommando appcmd und den entsprechenden Parametern ausführen, eine weitere wichtige Änderung, die Automatismen vereinfacht oder sogar erst möglich macht. Genauso komfortabel und effektiv ist die PowerShell, deren Benutzung für diesen Zweck in www.iis.net/extensions/PowerShell beschrieben wird.

Ein interessanter Ansatz, beispielsweise für den Einsatz bei Hosting-Providern oder anderen stark automatisierten Umgebungen, bildet die Anprogrammierbarkeit des IIS-Managements über eine .NET-Bibliothek. So lassen sich beliebig viele Applikationen und Applikationspools über eine C#-Applikation anlegen oder auch wieder löschen, wie Listing 1 zeigt.

»Fazit«

Als langjähriger LAMP-Administrator und PHP-Entwickler war ich dem IIS immer etwas skeptisch gegenüber gestanden. Und – zugegebener Maßen – habe ich es nicht einmal für notwendig gehalten, mir dessen Funktionen und Leistungsfähigkeiten auch nur anzusehen. Mit LAMP war ich glücklich und wollte keine Alternative. *httpd.conf* im Editor bearbeiten, *.vhost*-Dateien anlegen, Service auf der Kommandozeile neu starten, das alles wird zur Norm, wenn man nichts anderes kennt.

Ob es komfortabler geht, darüber habe ich mir herzlich wenig Gedanken gemacht. Und Komfort allein ist schließlich auch nicht viel wert, wenn Geschwindigkeit und Zuverlässigkeit dabei auf der Strecke bleiben. Nun aber, da die Frage nach Stabilität und Performanz der Vergangenheit angehört, gibt es keinen Grund auf den Komfort und die Zusatzfunktionen, die Microsofts IIS zu bieten hat, zu verzichten. Vielseitigkeit ist eine Tugend. Gerade in angespannten wirtschaftlichen Zeiten ist man froh, die vielleicht entscheidende Technologiekenntnis mehr im Portfolio zu haben.

Wenn dieser Wissensvorsprung dann auch noch Spaß macht und deren Anwendung leicht von der Hand geht, nehme ich das gerne an. PHP und IIS, schön, dass ihr es geschafft habt gute Freunde zu werden. [mb]

Listing 1

C#-Codebeispiel

```
using System;
using System.Collections.Generic;
using System.Text;
using Microsoft.Web.Administration;
namespace IIS7_createlotofsites {
    class Program {
        static void Main(string[] args)
        {
            ServerManager myServer = new ServerManager();
            Console.WriteLine("creating sites");
            ListSites(myServer);
            for (int i = 0; i < 1000; i++)
            {
                CreateSite(myServer, "site" + i.ToString());
            }
            myServer.CommitChanges();
            Console.WriteLine("after");
            ListSites(myServer);
            Console.WriteLine("delete the sites?");
            Console.ReadKey();
            for (int i = 0; i < 1000; i++)
            {
                DeleteSite(myServer, "site" + i.ToString());
            }
            myServer.CommitChanges();
            ListSites(myServer);
        }
        private static void CreateSite(ServerManager
        myServer, String siteName)
        {
        Site mySite = myServer.Sites.Add(siteName,
        "http", ":8080:", @"c:\temp");
            ApplicationPool myPool =
            myServer.ApplicationPools.Add(siteName + "-pool");
            myPool.ManagedPipelineMode = ManagedPipelineMode.Integrated;
            mySite.ApplicationDefaults.ApplicationPoolName = myPool.Name;
        }
        private static void DeleteSite(ServerManager myServer,
        String siteName)
        {
            try { myServer.Sites.Remove(myServer.Sites[siteName]);
                  myServer.ApplicationPools.Remove
                  (myServer.ApplicationPools[siteName + "-pool"]);
            }
            catch (Exception caught) {
                Console.WriteLine(caught.Message);
            }
        }
        private static void ListSites(ServerManager myServer) {
            foreach (Site mySite in myServer.Sites)
            {
                Console.WriteLine(mySite.Name);
            }
        }
    }
}
```