

Der Pinguin serviert

ASP.NET mit Mono

Auch wenn der IIS der meist eingesetzte Webserver für ASP.NET ist, heißt das nicht, dass ASP.NET nicht auch auf einem anderen Webserver lauffähig wäre. Neben Cassini, einem kleinen Webserver von Microsoft für Testzwecke, stehen mit dem Mono-Projekt gleich zwei alternative Webserver zur Verfügung: XSP als eigenständiges Programm und mod_mono zur Integration in Apache.

von Golo Haas

ASP.NET wird häufig als Konkurrenz zu PHP dargestellt, doch schaut man sich die Konzeption von ASP.NET einmal näher an, stellt man fest, dass dieser Vergleich hinkt. PHP ist im Gegensatz zu ASP.NET nicht dafür ausgelegt, eine Sprache für Webapplikationen im Enterprise-Umfeld zu sein, bei denen es auf maximale Skalierung und Wartbarkeit ankommt (obwohl natürlich auch mit PHP Enterprise-Lösungen möglich sind und existieren). Die wirkliche Konkurrenz von ASP.NET ist die Java 2 Enterprise Edition (kurz J2EE) von Sun. Auch wenn .NET von Microsoft als plattformunabhängig angekündigt wurde, war es dies im Gegensatz zu Java lange Zeit nicht wirklich – einfach deshalb, weil entsprechende Implementierungen für andere Betriebssysteme außer Windows fehlten.

Inzwischen gibt es mit dem Mono-Projekt [1] allerdings eine entsprechende

Umsetzung von .NET für Linux, Mac OS X und einige weitere Betriebssysteme (allerdings nicht von offizieller Stelle). Während Konsolenprogramme problemlos portierbar sind, ist dies bei Anwendungen, die auf Windows Forms oder ASP.NET aufbauen, nicht ganz so einfach. Nach einigen Anläufen kommt Mono mit Windows Forms inzwischen ganz passabel zurecht, und für das nächste große Release von Mono ist angekündigt, Windows Forms zu unterstützen. Wie jedoch sieht es denn mit ASP.NET aus?

IIS als Webserver ?

Der Haken bei ASP.NET ist letztlich, dass eine Anwendung nicht für sich alleine lauffähig ist, sondern einen Webserver benötigt, der die Anwendung ausführt und ausliefert. Der Referenzserver hierfür ist unter Windows ganz eindeutig der IIS (Internet Information Server) von Microsoft, der bei den verschiedenen Windows-Versionen mitgeliefert wird. Allerdings hat der IIS einige gravierende Nachteile [2].

Zum einen gibt es den IIS nicht als eigenständige Komponente, das heißt, eine bestimmte Version des IIS ist an eine bestimmte Windows-Version gekoppelt. Gibt es im Windows Server 2003 den IIS in Version 6.0, so muss Windows XP Professional mit dem IIS 5.1 und Windows 2000 gar mit der Version 5.0 vorlieb nehmen.

Möchte man auf den IIS 6.0 aktualisieren, so muss man gleich das komplette zu Grunde liegende Windows auf einen Windows Server 2003 migrieren, was mit entsprechenden Kosten und vor allem Aufwand verbunden ist.

Ein Grund, warum diese Aktualisierung aber dringend notwendig ist, ist, dass der IIS in früheren Versionen als 6.0 sehr viele Sicherheitslücken hat (für die es immerhin Patches gibt) und zudem in einer sehr unsicheren Konfiguration ausgeliefert wird. Er kann zwar abgesichert werden, jedoch steht der Aufwand hierfür in keinem Verhältnis zum Ergebnis. Mit dem IIS 6.0 hat Microsoft das Konzept des Webserver und seiner Standardkonfiguration deutlich überarbeitet, sodass diese Version bedenkenlos im Produktivbetrieb einsetzbar ist – auch wenn hier immer noch einige Einstellungen vorgenommen werden sollten.

Zudem ist ein weiterer Nachteil am IIS, dass er eben nicht plattformunabhängig verfügbar ist – man ist an die Windows-Plattform gebunden. Auch wenn man persönlich damit kein Problem hat, vielleicht widerspricht es der Firmenkultur, vielleicht passt es nicht in die Sicherheitsrichtlinie, vielleicht ist es auch einfach ein Kostenproblem. Es stellt sich also die Frage, wenn doch mit Mono .NET prinzipiell auch für Linux verfügbar ist, ob man ASP.NET

kurz & bündig

Inhalt

ASP.NET, Mono, Linux, Apache

Zusammenfassung

ASP.NET unter Linux mit Mono liefert eine nutzbare Kombination mit einigen Tücken

nicht auch mit anderen Betriebssystemen einsetzen kann?

XSP und mod_mono

Und tatsächlich, hier bringt Mono gleich zwei Lösungen mit: XSP und *mod_mono*. XSP ist ein eigenständiger ASP.NET-fähiger Webserver, der vollständig in .NET entwickelt wurde (und damit plattformunabhängig ist). Er wird in der Dokumentation von Mono prinzipiell als tauglich sowohl für den Entwicklungs- wie auch den Produktivbetrieb eingestuft. Allerdings sollte man im Produktivbetrieb beachten, dass XSP natürlich nicht über alle Funktionen eines großen, ausgewachsenen Webservers verfügt, wie es beispielsweise bei Load Balancing der Fall ist.

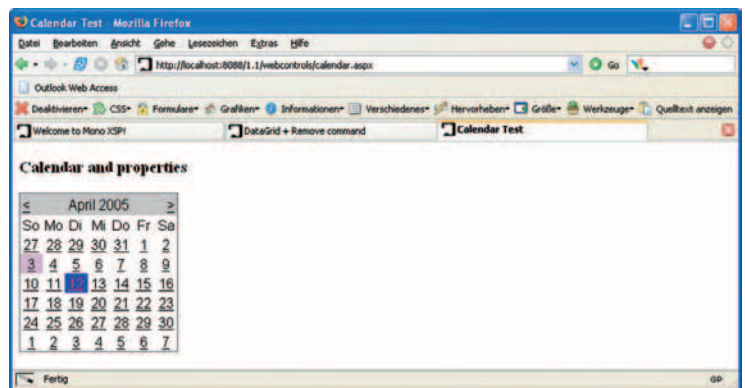
Da ein solcher Webserver unter Linux mit Apache [3] bereits existiert und nicht Sinn des Mono-Projektes sein kann, ebendiesen nachzuprogrammieren, bietet sich mit *mod_mono* eine gute Alternative an. *mod_mono* ist ein Plugin für den Apache (ein sogenanntes Modul), das Apache um die für ASP.NET benötigte Funktionalität erweitert. So kann Apache weiterhin als stabiler und ausgereifter Webserver verwendet werden und die Integration mit *mod_mono* erlaubt dann die Verarbeitung von in ASP.NET geschriebenen Applikationen.

Um so weit zu kommen, müssen Sie letztlich drei Hürden nehmen: Apache unter Linux zum Laufen bringen, Mono unter Linux zum Laufen bringen und die Integration von *mod_mono* in Apache hinbekommen.

Linux vorbereiten

Während für die auf Windows lauffähige Version von Mono ein komfortables Installationsprogramm bereit steht, ist man unter Linux auf eine Sammlung von RPM-Dateien angewiesen, die teilweise wiederum von weiteren Paketen abhängen. Novell arbeitet derzeit daran, Mono in zukünftige Versionen von SuSE Linux zu integrieren, momentan erfordert die Installation allerdings noch einiges an händischem Aufwand, um als Entwickler zumindest annähernd auf den gleichen Komfort wie unter Windows zurückgreifen zu können. Dennoch stellt Novell SuSE Linux 9.1 derzeit eine sehr gute Basis dar, um Mono zu nutzen; eine Installation auf anderen Distributionen sollte allerdings, ähnlich

Abb. 1:
Calendar and properties



wie im Folgenden beschrieben, auch möglich sein.

Außer einer vollständigen Installation von Mono, also einschließlich Windows Forms, Gtk#, ASP.NET, ADO.NET, der integrierten Entwicklungsumgebung MonoDevelop, dem Webserver XSP und einigen nützlichen Werkzeugen, werden zusätzlich noch der Webserver Apache, die Datenbanken MySQL und PostgreSQL sowie das Versionierungswerkzeug Subversion installiert. Ausgehend von einer Minimalinstallation von Novell SuSE Linux 9.1 werden für eine derartige Installation von Mono folgende Pakete benötigt, die mittels YaST2 bequem über eine grafische Oberfläche installiert werden können, sofern sie nicht bereits vorhanden sind:

- apache2
- compat
- findutils-locate
- ifnt*
- libmng
- make
- man-pages
- mysql
- mysql-client
- pkgconfig
- postgresql
- postgresql-server
- subversion
- unzip
- XFree86
- XFree86-fonts-*

Mit der Installation dieser Pakete sind alle Vorbereitungen getroffen, um sich an die eigentliche Installation von Mono heranzuwagen. Eine Möglichkeit, Mono immer aktuell zu halten und mit Updates zu versorgen, ist, regelmäßig auf der Webseite des Projektes vorbeizuschauen und im Falle eines Falles aktualisierte Pakete her-

unterzuladen und einzuspielen. Eine deutlich bequemere Alternative dazu bietet das Werkzeug *Red Carpet* an, mit dessen Hilfe nach neuen Updates gesucht werden kann und diese direkt installiert werden können, wobei sogar Abhängigkeiten automatisch aufgelöst werden. Da sich dieser Vorgang zudem mittels eines Cronjobs automatisieren lässt, erhält man ein System, das sich selbst auf dem aktuellen Stand hält.

Um Red Carpet zu installieren, müssen lediglich eine Hand voll RPM-Dateien von der Webseite des Mono-Projektes heruntergeladen werden, die dann als Benutzer *root* mittels

```
rpm -Uvh *.rpm
```

installiert werden können. Unter Umständen kann es passieren, dass einige Abhängigkeiten von anderen Paketen, die zu Novell SuSE Linux 9.1 gehören, nicht aufgelöst werden können. Sollte dies der Fall sein, kann man diese über YaST2 allerdings problemlos nachinstallieren. Als Nächstes muss das System neu gestartet werden, damit der Red-Carpet-Dämon aktiv wird und angesprochen werden kann. Falls ein Proxy eingesetzt wird, muss dieser auch noch konfiguriert werden, indem man

```
rug set proxy-url [Protokoll]://[Url oder IP-Adresse]:[Port]
```

ausführt. Auch nach diesem Schritt muss das System neu gestartet werden, damit die Änderungen wirksam werden. Bevor Red Carpet nun ausgeführt werden kann, um Mono zu installieren, muss noch eine Konfigurationsdatei des Apache kopiert werden, die in einem anderen als dem von Novell SuSE Linux 9.1 verwendeten Ordner erwartet wird:

```
cp/etc/apache2/sysconfig.d/loadmodule.conf/etc/
    apache2/httpd2-prefork.conf
```

Mono und XSP installieren

Red Carpet arbeitet kanalorientiert, wobei ein Kanal einem Thema entspricht. So gibt es beispielsweise einen Kanal mit Updates für Novell SuSE Linux 9.1, einen für den Kernel von Linux und schließlich eben auch einen für Mono. Aus diesen Kanälen lassen sich einzelne Dateien herunterladen, man kann den Kanal aber auch als ganzes beziehen. Zunächst werden also die Kanäle für Novell SuSE Linux 9.1, den Kernel, Red Carpet an sich und Mono dem System bekannt gemacht, und danach eine vollständige Installation von Mono durchgeführt:

```
rug sub mono
rug sub redcarpet2
rug sub suse-91-i586
rug sub suse-91-i586-kernel
rug in --entire-channel mono
rug up
```

Der letzte Befehl, *rug up*, führt ein Update aller installierten Pakete aus, die in einem der Kanäle enthalten sind und für die neuere Versionen vorliegen. Lediglich dieser Befehl muss auch mit Cron eingerichtet werden, um regelmäßig das System zu aktualisieren. Damit ist Mono bereits installiert, wie ein Aufruf des Compilers für C# mittels *mcs* oder der Runtime mittels *mono* zeigt.

Auch XSP ist damit bereits installiert, eine weitere Konfiguration des Programms ist nicht notwendig. Um den Aufruf allerdings ein wenig zu vereinfachen, legt man am besten eine Datei namens *xsp* im Ordner */bin/* an, fügt die Zeilen

```
#!/bin/bash
mono /usr/bin/xsp.exe $*
```

ein und gibt mittels

```
chmod a+x /bin/xsp
```

jedem Benutzer Ausführungsrechte. Nun lässt sich XSP durch den Aufruf seines Namens von jedem beliebigen Verzeichnis aus starten, wobei das derzeitige Verzeichnis zum Wurzelverzeichnis für den Webserver wird. Ein Tastendruck auf *Enter* beendet XSP wieder. Da diese Funktionalität auf manchen Systemen aber nicht einwand-

frei arbeitet, muss man sich hier mit der Tastenkombination *Strg+C* behelfen, um XSP abzubrechen. Standardmäßig arbeitet XSP auf Port 8080, sodass sich nach dessen Start der Webserver über *http://localhost:8080* bereits ansprechen lässt.

mod_mono in Apache einbinden

Vielleicht ist Ihnen beim Herunterladen der RPM-Dateien aufgefallen, dass bei *mod_mono* der Hinweis stand, dass XSP als Basis vorausgesetzt wird. Das liegt daran, dass *mod_mono* letztlich nur ein Wrapper für XSP ist, der als Modul in Apache eingebunden werden kann. Zunächst soll-

mod_mono als Wrapper für XSP

te man allerdings, analog zu Windows, einen Ordner anlegen, der als Document Root für ASP.NET-Applikationen dient, wobei der Name natürlich beliebig ist und auch unterhalb des normalen Wurzelverzeichnis des Apache liegen könnte:

```
mkdir /Inetpub
mkdir /Inetpub/wwwroot
```

Da *mod_mono* in der Version 1.0.1 noch nicht im Mono-Kanal von Red Carpet enthalten ist, muss diese Datei gesondert heruntergeladen und anschließend mit folgendem Befehl installiert werden:

```
rpm -Uvh mod_mono-1.0.1-0.ximian.9.0.i586.rpm
```

Als nächstes muss Apache so konfiguriert werden, dass das Modul eingebunden wird. Dazu öffnen Sie die Datei */etc/apache2/httpd.conf*, in der sich die zentrale Konfiguration von Apache befindet, und suchen die Zeile, welche den *Directory-Index* definiert. Die Direktive *Directory-Index* definiert, welche Datei ausgeliefert werden soll, wenn eine Webseite aufgerufen wird, ohne dass explizit eine Datei angegeben wird. Unter anderem dürfte sich hier der Eintrag *index.html* finden. Da sich unter ASP.NET der Dateiname

```
Default.aspx
```

für die Startseite einer Webanwendung eingebürgert hat, fügt man schließlich die-

sen als neuen Eintrag hinzu. Zusätzlich muss in der Datei */etc/apache2/default-server.conf* das Modul *mod_mono* eingebunden werden, indem man dieser Datei die Zeilen

```
LoadModule mono_module (usr/lib/apache2/libmod_
    mono.so
MonoApplicationsConfigDir /Inetpub/wwwroot
```

anhängt. Außerdem müssen auf den Ordner, in dem die ASP.NET-Anwendungen abgelegt werden, sowie alle untergeordneten Ordner, noch entsprechende Lese- und Schreibrechte mittels

```
chmod 755 /Inetpub -R
```

gesetzt werden. Damit Apache nach jedem Neustart des Systems gestartet wird, wird innerhalb von YaST2 in der Kategorie System der Runlevel-Editor aufgerufen, in dem überprüft werden kann, ob der Dämon *httpd* aktiviert ist, falls nicht, kann es hier entsprechend nachgeholt werden. Als nächstes muss der Inhalt der Datei */usr/bin/mod-mono-server* mit den folgenden Zeilen ersetzt werden, um das Laden von *mod_mono* in das Starten von Apache zu integrieren:

```
#!/bin/bash
echo Starting mod-mono-server ...
mono /usr/bin/mod-mono-server.exe --root /Inetpub/
    wwwroot --appconfigdir /Inetpub/
    wwwroot --nonstop &
sleep 3
echo Setting up mod-mono-server socket ...
chmod a+rw /tmp/mod_mono_server
```

Dadurch wird sichergestellt, dass nach dem Starten des Moduls *mod_mono* auf den Socket zugegriffen werden kann. Die Pause von drei Sekunden ist notwendig, da *mod_mono* einen kleinen Augenblick benötigt, bis es gestartet ist. Als letzter Schritt der Konfiguration von ASP.NET für Apache wird dieses Skript noch in das Start- beziehungsweise Stopp-Skript von Apache integriert, indem man die Datei */etc/init.d/apache2* bearbeitet. Zunächst muss der Abschnitt gesucht werden, in dem Apache gestartet wird. An dessen Ende – also direkt vor den beiden Semikolons – wird die Zeile

```
mod-mono-server
```

eingefügt, ebenso wird am Ende des Abschnitts, in dem Apache gestoppt wird, die Zeile

```
killall mono
```

eingefügt, um `mod_mono` wieder zu beenden. Damit ist ASP.NET vollständig konfiguriert und Anwendungen können unterhalb von `/Inetpub/wwwroot` abgelegt und konfiguriert werden. Zudem sind XSP, die Compiler, die Runtime und die Entwicklungsumgebung MonoDevelop fertig installiert und einsatzbereit.

ASP.NET-Anwendungen konfigurieren

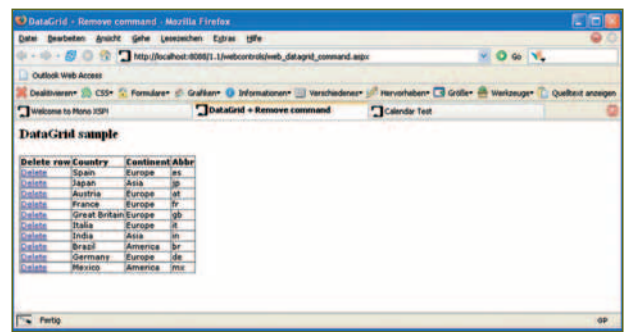
Unter Visual Studio .NET wird das Veröffentlichen einer Webanwendung im Hintergrund mit einem Klick erledigt. Auch falls im Nachhinein noch einmal eine Version händisch veröffentlicht werden sollte, ist dies beim IIS nur ein `xcopy` entfernt. In der Regel kopiert man die Dateien in das entsprechende Verzeichnis, wodurch beim nächsten Aufruf im Browser bereits die neue Version ausgeliefert wird. Dass die Webanwendung im IIS allerdings initial erst einmal konfiguriert werden muss, verbirgt Visual Studio .NET vollständig vor dem Anwender.

Unter Apache ist diese erstmalige Konfiguration derzeit nur händisch möglich, allerdings beschränkt es sich auf einige wenige Anpassungen in ein paar Dateien, sodass das Aufsetzen einer neuen Webanwendung in wenigen Minuten erledigt ist. Zunächst sollte man für jede Anwendung ein eigenes Verzeichnis unterhalb von `/Inetpub/wwwroot` anlegen, wobei man darauf achten muss, im Namen keine nicht-alphanumerischen Zeichen zu verwenden, da `mod_mono` sonst nur Fehlermeldungen ausgibt. Als nächstes wird die bereits kompilierte Anwendung einschließlich der `.aspx`-Dateien und des `bin`-Ordners in dieses Verzeichnis kopiert, wobei wiederum die Zugriffsrechte mittels

```
chmod 755 /Inetpub -R
```

gesetzt werden müssen. Der Besitzer und die Gruppe, die für die jeweiligen Dateien und Ordner eingetragen sind, sind für `mod_mono` uninteressant, so lange die Berechtigungsmaske 755 korrekt gesetzt wurde.

Abb. 2: DataGrid-Beispiel



Als nächstes muss die neue Webanwendung für Apache bekannt gemacht und auf einen entsprechenden virtuellen Pfad gebunden werden. Dazu öffnet man die Datei `/etc/apache2/default-server.conf` und fügt die folgenden Zeilen an, wobei die jeweiligen Pfade natürlich an die entsprechende Anwendung angepasst werden müssen:

```
Alias /VirtualPath /Inetpub/wwwroot/PhysicalPath
<Location /VirtualPath>
  Order allow,deny
  Allow from all
  SetHandler mono
</Location>
```

Damit weiß Apache nun, wie er mit einem entsprechenden Request an den virtuellen Pfad umgehen soll, nämlich an den Handler `mono` weiterleiten, der hier stellvertretend für das Modul `mod_mono` steht. Allerdings muss auch diesem noch mitgeteilt werden, wie es mit dem Request umgehen soll, wozu im Ordner der neuen Anwendung eine Datei erstellt wird, deren Name dem Namen des Ordners der Anwendung, ergänzt um die Erweiterung `.webapp`, entspricht. Dabei ist zu beachten, dass die Groß- und Kleinschreibung des Dateinamens korrekt eingehalten wird, da `mod_mono` sonst nicht auf die Datei zugreifen kann. Diese Datei ist eine XML-Datei und enthält folgende Zeilen:

```
<?xml version="1.0" encoding="utf-8" ?>
<apps>
  <web-application>
    <name>ApplicationName</name>
    <vhost>ServerName</vhost>
    <vport>ServerPort</vport>
    <vpath>/VirtualPath</vpath>
    <path>/Inetpub/wwwroot/PhysicalPath</path>
  </web-application>
</apps>
```

Sobald Apache neu gestartet wurde, ist die Webanwendung über den konfigurierten virtuellen Pfad erreichbar.

Ausblick

Ein wesentlicher Nachteil dieser Lösung gegenüber dem IIS soll hier jedoch nicht verschwiegen werden. Der IIS unterstützt nämlich eine Technik namens XCOPY Hot Deployment, das bedeutet, im laufenden Betrieb kann eine Datei einer Webanwendung durch eine neue Version ersetzt werden, ohne dass die Webseite angehalten und wieder gestartet werden muss.

Dies wird derzeit leider weder von XSP noch von `mod_mono` unterstützt, sodass nach jeder Änderung der Webanwendung Apache neu gestartet werden muss, damit `mod_mono` neu initialisiert wird. Aktualisiert man lediglich die Dateien, ohne Apache neu zu starten, so wird die Webanwendung über kurz oder lang nicht mehr fehlerfrei laufen und der Benutzer bekommt Fehlermeldungen angezeigt.

Bleibt zu hoffen, dass dies in einem zukünftigen Release von Mono behoben sein wird. Sieht man von dieser Einschränkung ab, ist Apache mit `mod_mono` aber schon eine sehr ausgereifte Alternative zum IIS. ●

Golo Haas ist Softwarearchitekt und -entwickler bei der Kheops GmbH in Kaiserslautern und entwickelt mit .NET und Visual FoxPro. Er ist der Autor von „Guide to C#“ (www.guidetocsharp.de) und Leiter der .NET Usergroup Kaiserslautern (www.netug-kl.de). Sie können ihn unter webmaster@golohaas.de erreichen.

Links & Literatur

- [1] www.mono-project.com
- [2] www.netug-kl.de, Vortrag vom 14. Februar 2005
- [3] httpd.apache.org