Foto: Mathias Vietmeier

PHP auf Windows Azure

Interoperable Wolke

Cloud Computing ist inzwischen für jeden IT-Interessierten ein Begriff. Was steckt hinter der Windows Azure Plattform und warum will Microsoft gerade hier so gut Freund sein mit PHP? Von Jan Schenk

AUF EINEN BLICK

- Die Windows Azure-Plattform bietet eine Basis für Anwendungen, die direkt in der Cloud ausgeführt werden.
- Der Workshop erläutert die Details der Plattform und zeigt, wie man darauf PHP-Applikationen realisieren kann.

Cloud Computing - also die Bereitstellung von IT-Funktionen als Dienstleistung in Form einer Software, einer Plattform oder einer Infrastruktur - macht große Versprechen. Zu nennen wäre da Skalierbarkeit in nie da gewesenen Dimensionen, enorme Flexibilität, Kosteneinsparungen gegenüber klassischen Hardware-Investitionsmodellen, maximale Ausfallsicherheit. Und nicht zu vergessen GreenIT, also die Hoffnung, Kosten reduzieren und gleichzeitig ein besseres Gewissen haben zu können.

Cloud Computing gilt als nächste Evolutionsstufe der IT. War in den Fünfziger bis in die Siebziger Jahre der Großrechner oder Mainframe das dominierende Modell, entwickelte sich dieses über die nächsten zehn Jahre zum Client-Server-Prinzip, in dem Rechenaufgaben vornehmlich vom Client erledigt wurden. Seit den Neunzigern dominiert das Internet die IT-Landschaft, und wir spüren einen unaufhaltbaren Trend zu einer serviceorientierten Architektur. Der nächste logische Schritt bedeutet, dass wir uns einem Modell nähern, das Rechen- und Speicheraufgaben wieder zentralisiert. Diesmal allerdings mit anderen Anforderungen (unkalkulierbare Lastspitzen aufgrund eines globalen Nutzerkreises) und anderen Voraussetzungen (Hochgeschwindigkeitsanbindungen an das Internet).

Die Windows Azure Plattform bietet zum einen eine administrationsfreie Basis für Anwendungen, die direkt in der Cloud ausgeführt werden. Zum anderen aber ist Microsoft ein Geniestreich bezüglich Interoperabilität gelungen. Auf der Professional Developers Conference im November 2009 wurde gezeigt, wie die Blogsoftware Wordpress in Windows Azure in einer Apache/MySQL/PHP-Umgebung installiert und ausgeführt wird. Das erstaunliche hierbei ist, dass die Windows Azure Platform von Beginn an dafür konzipiert wurde, Microsoft-fremde Bausteine, in diesem Fall der Apache Webserver und der MySQL Datenbankdienst, zu integrieren. Der einzige Wermutstropfen ist, dass jegliche Administration dieser Bestandteile wieder in der Verantwortung des Kunden, also des Betreibers der Anwendung und nicht dem

Betreiber der Cloud, liegt. Ein Aspekt der Windows Azure Plattform - die Administrationslosigkeit - geht dadurch wieder verloren. Das ist, versetzt man sich an Microsofts Position, nur zu gut nachzuvollziehen, stünde der Softwarehersteller doch sonst in der Schuld für den Drittanbieter einer Open-Source-Software Support leisten zu müssen. Weiterhin keine Sorgen machen muss man sich aber um das Betriebssystem und dessen Vitalität und Sicherheit. Wer damit leben kann, seine Dienstkomponenten selbst zu warten, für den bietet Windows Azure hinsichtlich Interoperabilität eine flexible Basis mit all den anderen zu Beginn genannten Versprechen.

Den vollen Umfang der Administrationsfreiheit bietet Microsofts Cloud Computing mit dem eigenen .NET-Stack: WISA, also Windows, IIS, SQL Azure und ASP.NET. Und auch mit PHP in Form eines WISP-Modells zeigt sich Windows Azure von seiner besten Seite.

Da Windows Azure im Kern auf den gleichen Techniken basiert wie Windows Server 2008 und der Internet Information Service (IIS), ist die Unterstützung von PHP, insbesondere PHP 5.3, dank FastCGI die beste die es jemals gab. Sowohl in Sachen Stabilität, als auch hinsichtlich der Performance. So ist der IIS mit PHP 5.3, je nach Anwendungszweck, genauso schnell oder sogar schneller als lighttp oder Apache.

Das Konzept der Plattform

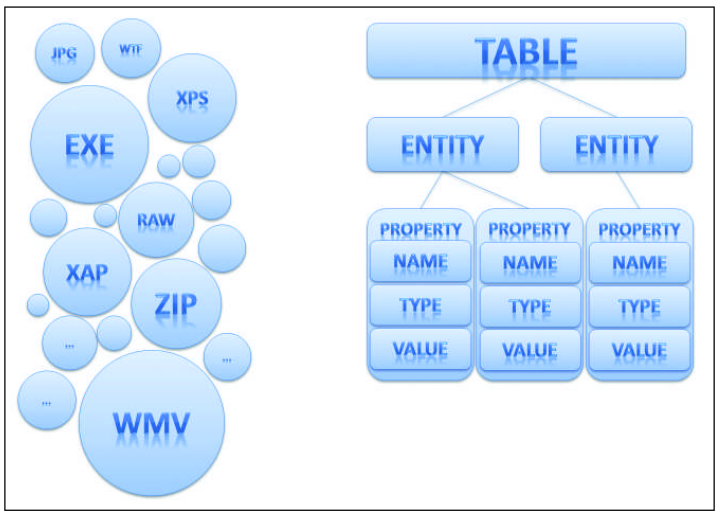
Die Windows Azure-Plattform ist modular aus drei großen Komponenten aufgebaut: Windows Azure, SQL Azure und die appFabric. Kurz erklärt ist die Aufgabe der Komponente Windows Azure, die für das Rechnen und Speichern zuständig ist. SQL Azure ist ein relationaler Datenbankdienst, der Datenbanken bereitstellt und automatisch skaliert. Und die Windows Azure Plattform appFabric, wie der volle Name lautet, kümmert sich um die sichere Kommunikation zwischen Anwendungen innerhalb und ausserhalb der Wolke nach dem Claim-Token-Verfahren. Nähere Informationen dazu finden Sie unter anderem in Holger Sirtls Blog auf blogs.msdn.com/hsirtl/.

Windows Azure, die Komponente, die die Applikation beheimatet und ausführt und damit das Herzstück von Microsofts Cloud Computing, unterteilt sich wiederum in zwei große Komponenten: Computing, also Rechenleistung, und Storage, der Datenspeicher.

Befassen wir zunächst mit der Ausführungsschicht, dem Computing. Das Programmiermodell für die Computing-Komponente folgt einem Rollenkonzept. So gibt es verschiedene Rollen, in die man die einzelnen Teile der Anwendung logisch aufteilt. Prinzipiell sind diese Rollen nur Vorlagen, in deren Konfigurationsdateien schon bestimmte Werte gesetzt sind. Zur Verfügung stehen die Vorlagen Web-Rolle, MVC Web-Rolle, Worker-Rolle, FastCGI Web-Rolle und WCF Service Web-Rolle. Web-Rollen sind die wohl einfachste Form der Rolle. Eine klassische Webseite würde in einer einzigen Web-Rolle untergebracht werden und bräuchte in den meisten Fällen keine anderen Rollen. Diese Rollen können unabhängig voneinander in Instanzen skaliert werden, wodurch ein flexibles System entsteht, das präzise Lastanforderungen erfüllen kann.

LISTING 1: MICROSOFT_WINDOWSAZURE_STORAGE_BLOB

```
<?php
require_once 'Microsoft/WindowsAzure/Storage/Blob.php';
// Verbinden zum Development Storage
$storageClient = new Microsoft_WindowsAzure_Storage_Blob();
// Prüfe, ob der Container bereits existiert
$containerExists = $storageClient->containerExists("beispiel");
if(!$containerExists)
{
        // Erstellen eines BLOB Containers namens "beispiel"
        $storageClient->createContainer("beispiel");
        print("Container erstellt!<br />");
}
else print("Container existiert bereits!<br />");
// Setzen von oeffentlichen Leserechten
$storageClient->setContainerAcl("beispiel", Microsoft_WindowsAzure_Sto-
rage_Blob::ACL_PUBLIC);
// Hochladen der php.ini Datei
$storageClient->putBlob("beispiel", "php.ini", "C:/Program Files
(x86)/PHP/php.ini");
// Abrufen der mit dem BLOB verbundenen Details
$result = $storageClient->getBlobInstance("beispiel", "php.ini");
// Zugriff auf das BLOB per Browser (nur bei oeff. Leserechten)
echo "<br />Hier k&ouml;nnen Sie das <a href=" . $result->url . ">BLOB
herunterladen</a>.";
?>
```

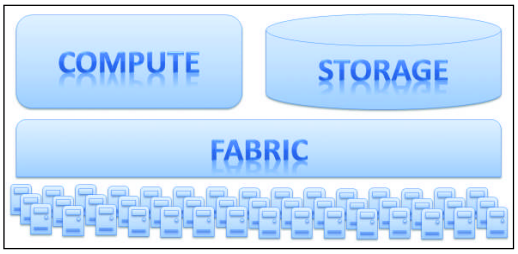


Mircosoft gibt sich mit der PHP-Unterstützung inzwischen sehr viel Mühe (Bild 1)

LISTING 2: WEB.ROLECONFIG

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <system.webServer>
    <fastCgi>
      <application fullPath="%Role-
Root%\approot\php-5.3.1\php-cgi.exe"
arguments="-c php.ini" />
    </fastCgi>
  </system.webServer>
</configuration>
```

Die zweite Komponente, Storage, ist auch noch einmal unterteilt in die Modelle BLOBs, Tables und Queues. BLOBs sind genau das, was wir uns darunter vorstellen, binary large objects, oder einfach Binärobjekte, die in einer flachen Hierarchie angeordnet sind. Tables bezeihungsweise Tabellen hingegen verleiten oftmals zu der falschen Schlussfolgerung, es würde sich hierbei um Tabellen im relationalen Sinne handeln, wie wir sie aus Datenbanken kennen. Tabellen sind in Windows Azure nichts anderes als Key-Value-Pairs (Schlüssel-Wert-Paare), die in einer hierarchischen Ordnung von Entitäten und Eigenschaften organisiert werden (Bild 1).



Die grafische Administrationsoberfläche des IIS erlaubt schnelle Änderungen an der Server-Konfiguration (Bild 2)

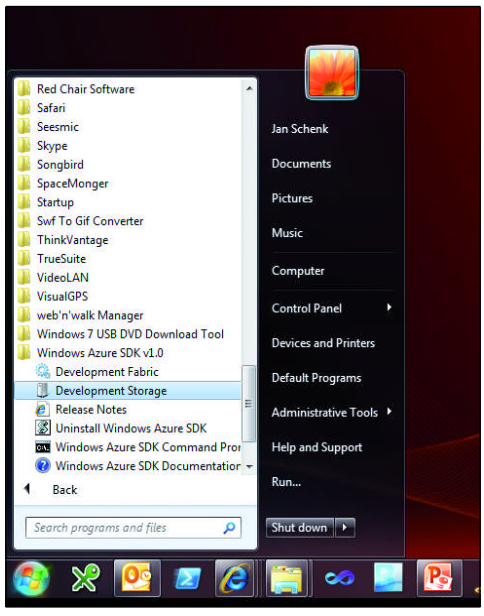
Gerade weil dieses Modell der Tables keinerlei Relationalität integriert, ist es ohne Aufwand massiv skalierbar. Der enorme Preisunterschied zu relationalen Datenbanken in der Cloud lässt einen strategisch überlegen, ob für alle gespeicherten Daten tatsächlich Relationalität notwendig ist. Der Großteil der heute in relationalen Datenbanken abgelegten Daten hat keinen zwingenden Grund relational gespeichert zu sein. Eine einfachere Struktur, wie die der Windows Azure Tables, ist meist mehr als genug. Vereinzelt trifft man sogar noch Anwendungen, die klassische BLOBs wie Videos und Bilder in Datenbanken ablegen. Sowohl Performance als auch Kosten betreffend ist das nicht nur in Windows Azure ein teurer und genauso überflüssiger Luxus.

Die beiden Komponenten, Computing und Storage, werden durch die sogenannte Fabric verwaltet. Die Fabric ist das Management von Windows Azure. Sie kümmert sich um Skalierung und Überwachung der Anwendungen. Ausserdem ist die Fabric die Abstraktionsschicht über der unglaublichen Menge an Hardware, die in Microsofts Rechenzentren steht (Bild 2). Für ein aussagekräftiges Beispiel der Rollenkonzeptes dient in meinen Vorträgen über Windows Azure oft das Videoportal Youtube. Dank dessen Bekanntheitsgrad ist es als Beispiel ideal. Ein typisches Szenario dort kann man so umschreiben: Ein User lädt ein Video hoch, fügt Metadaten hinzu und dieses Video wird verarbeitet. Kurze Zeit später steht es allen Benutzern der Videoplattform zur Ansicht zur Verfügung.

Überträgt man diese Aufgabe in das Windows Azure Konzept, hätten wir folgende Verwendung der einzelnen Komponenten: Ein Web-Frontend in einer Web-Rolle, einen Transkodierdienst im Hintergrund in Form einer Worker-Rolle, BLOBs in denen die verschiedenen Versionen des Videos abgelegt werden, und Tables, die die Metadaten des Videos beinhalten. Wie kommt nun der Trankodierdienst an seine Arbeit. Dazu dienen die Windows Azure Queues. Queues sind Warteschlangen, in die das Web-Frontend eine Aufgabe einreiht, damit eine Worker-Rolle sich diese aus der Queue, wenn sie ihren vorhergehenden Job abgearbeitet hat, abholt. Ein einfaches Konzept, das es uns aber erlaubt, die einzelnen Komponenten asynchron arbeiten zu lassen und unabhängig voneinander zu skalieren. So kann man je nach Bedarf die Instanzen des Web-Frontends erhöhen, oder aber die des Hintergrunddienstes, der Worker-Rolle. Geht die Last zurück, ist ein Drosseln der bereitgestellten Ressourcen jederzeit und mit unmittelbarer Auswirkung möglich.

Programmieren für Windows Azure

Es empfiehlt sich, sowohl das Windows Azure SDK, als auch die Windows Azure Tools für die Entwicklung von Windows Azure-Anwendungen einzusetzen. An dieser Stelle sei erwähnt, dass es die Windows Azure-Tools nicht nur für Visual Studio (www.microsoft.com/germa-



Der IIS-Webserver lässt sich sehr einfach über Erweiterungen mit zusätzlichen Fähigkeiten ausrüsten (Bild 3)

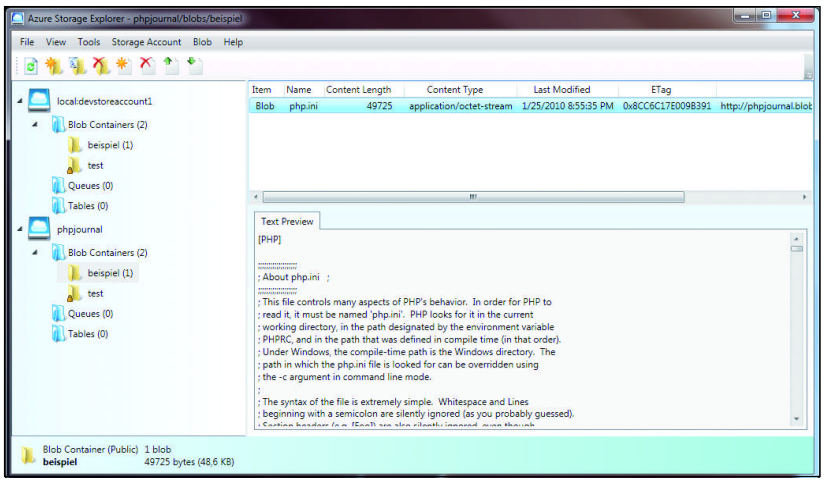
ny/visualstudio/) und das kostenlose Visual Web Developer Express Edition (www.microsoft.com/germany/express/) gibt, sondern unter windowsazure4e.org auch für Eclipse zur Verfügung stehen. Dort gibt es auch eine ausführliche Anleitung, wie man die Tools in Eclipse integriert. Ergänzend sei hinzugefügt, dass man Eclipse zur Installation der Windows Azure-Tools unbedingt als Administrator starten sollte. Das verschweigt die Anleitung leider.

Hat man seine Entwicklungsumgebung soweit vorbereitet, steht mit dem zuvor installierten Windows Azure SDK eine Laufzeitumgebung auf der Entwicklermaschine bereit (Bild 3). Diese simuliert sowohl die Windows Azure Fabric, also die Ausführungsschicht, als auch das Windows Azure Storage, die Speicherschicht. Lokal heißen diese beiden Umgebungen dann Development Fabric und Development Storage. Diese starten automatisch, wenn man Projekte in einer der unterstützten Entwicklungsumgebungen lädt, sie lassen sich aber auch unabhängig über das Programmmenü aufrufen, Das ist hilfreich, wenn man, wie im nachfolgenden Beispiel, nur die Speicherfunktionalitäten benötigt und die zugreifende PHP-Applikation nicht in einem Windows Azure Projekt gehostet ist, weil sie beispielsweise auf einem Server im eigenen Rechenzentrum läuft.

Ein Vorzeigeprojekt zur Interoperabilität von Microsofts Cloud Computing-Plattform ist das Windows Azure PHP Software Development Kit (kurz Azure PHP SDK), das von der belgischen Firma RealDolmen unter BSD-Lizenz auf codeplex (www.codeplex.com/phpazure) veröffentlicht wurde. Das Azure PHP SDK ermöglicht es Anwendungen, auf Funktionen in Windows Azure zu zu greifen. Mit Hilfe einer Wrapper-Klasse wird eine Abstraktionsschicht geschaffen, mit Hilfe derer man die Storage-Funktionalitäten von Windows Azure auf einfache Art und Weise nutzen kann. Dies muss nicht aus der Wolke heraus statt finden, die PHP-Anwendung kann auf einem beliebigen Server laufen. Außerdem stellt das Azure PHP SDK Hilfsklassen für die Authentifizierung und Autorisierung sowie für den HTTP-Transport, REST und Fehlermanagement zur Verfügung.

Um ein Objekt, beispielsweise User-Generated Content, nun in den BLOB-Storage in Windows Azure hoch zu laden und wieder abzurufen, reichen ein paar wenige Zeilen PHP-Code, die das SDK, das in den include-Pfad von PHP kopiert wurde, referenzieren (Listing 1).

Die Authentifizierung erfolgt in der lokalen Entwicklungsumgebung von Windows Azure, dem Development Storage, automatisch. Um Operationen im echten Cloud Storage durchzuführen, ruft man den Klassenkonstruktor gleich am Anfang mit speziellen Parametern auf:



Die IIS-Erweiterung UrlRewrite erlaubt das Konvertieren von htaccess-Rewrite-Regeln (Bild 4)

```
$storageClient = new Microsoft_Wind-
owsAzure_Storage_Blob(
Microsoft_WindowsAzure_Storage::URL_CL
OUD_BLOB, 'phpjournal', 'derPrimaryAc-
cessKey5RO3gpaQAXmIsOb5zkWmZv+0jnki-
iPRFp5VmaK8mfq/g==' );
```

Damit das auch funktionieren kann muss man einen Account unter windows.azure.com mit einer Live-ID registriert haben. Um Entwicklern auch den richtigen Eindruck von der echten Cloud-Umgebung zu vermitteln gibt es vorerst bis zum 31. Juli eine kostenlose Variante, die monatlich 25 Stunden Rechenzeit und 500 MByte Speicherplatz bereit hält. Eigentlich zu wenig, um damit zu arbeiten, aber genug, um sich mit der Plattform vertraut zu machen. Details dazu können auf www.microsoft.com/windowsazure/offers/ unter Introductory Special abgerufen werden. Wer regelmässig in beiden Welten (PHP und .NET) unterwegs ist, der kennt vielleicht auch die kostenpflichtige MSDN Premium Subscription. In deren Rahmen gibt es derzeit ein Inklusivprogramm rund um Windows Azure, das wesentlich mehr Rechenzeit, Speicherplatz und Traffic beinhaltet, ohne dass dafür Kosten anfallen. Weitere Informationen zur MSDN Subscription finden Sie im Infokasten. s

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.

E blogs.msdn.com/jansche/

LINKS ZUM THEMA

Azure Information Day am 16. März 2010 in München
- www.event-team.com/events/azure/Anmeldung.aspx

MSDN-Subscription
- www.microsoft.de/visualstudio

Holger Sirtls Blog
- blogs.msdn.com/hsirtl/

Visual Studio
- www.microsoft.com/germany/visualstudio/

Visual Web Developer Express Edition
- www.microsoft.com/germany/express/

Windows Azure Tools für Eclipse
- windowsazure4e.org

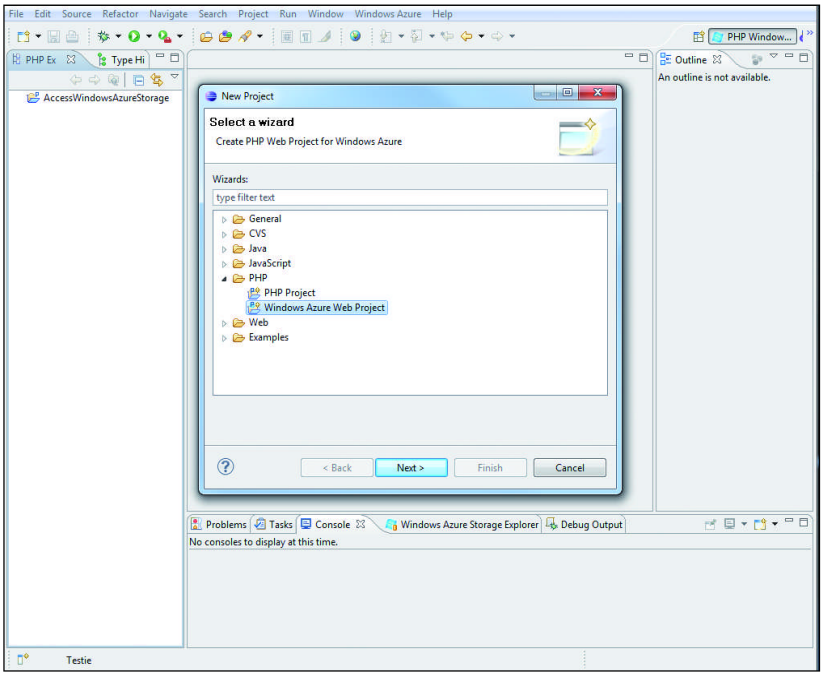
Azure PHP SDK
- www.codeplex.com/phpazure

Azure Storage Explorer
- azurestorageexplorer.codeplex.com

Ein weiteres Projekt um Windows Azure ist die ebenfalls auf codeplex.com veröffentlichte Azure Storage Explorer Desktopanwendung (azurestorageexplorer.codeplex.com). Mit dieser Anwendung lassen sich sowohl Inhalte in der Azure Storage Komponente, als auch im Development Storage verwalten. Gerade bei der Entwicklung von Azure-Applikationen, die Speicherplatz in der Cloud verwenden, ist der Azure Storage Explorer ein essentielles Werkzeug, um schnell und problemlos BLOBs, Tables und Queues zu überprüfen oder händisch zu bearbeiten (Bild 4).

Im nächsten Schritt bringen wir eine minimale Beispielapplikation in die Cloud. Eclipse startet man derzeit am Besten als Administrator, um einigen Schwierigkeiten mit der Azure Entwicklungslaufzeit, die Administratorrechte benötigt, aus dem Weg zu gehen. Im Assistenten für neue Projekte befindet sich nun ein neuer Eintrag im PHP-Menü: Windows Azure Web Project. Wählt man diesen aus, so öffnet sich ein weiteres Fenster, um das Projekt zu konfigurieren (Bild 5).

Wir belassen die Einstellungen auf den voreingestellten Werten, vergeben einen Projektnamen und legen das Projekt an. Eclipse erstellt nun automatisch zwei Ordner, <ProjektName> und <ProjektName>_WebRole. Im folgenden interessiert uns lediglich der Ordner mit der Namenserweiterung _WebRole. Dort legen wir zwei zusätzliche Dateien an, eine Web.config und eine Web.roleconfig. Diese sind Konfigurationsdateien für Windows Azure, mit denen wir ein bestimmtes Verhalten erzwingen, ähnlich der .htaccess-Datei auf Apache Webservern. Während die web.roleconfig einen trivialen Inhalt hat (Listing 2) ist die web.config im Normalfall sehr viel ausführlicher. Ein funktionierendes Beispiel befindet sich in Listing 3, diese stark reduzierte Version ist aber nicht für den Produktivbetrieb zu verwenden. Ein ausführlicheres Beispiel einer web.config-Datei finden Sie im Web unter der Adresse msdn.microsoft.com/dede/library/k8x4ket8(VS.80).aspx.



Jeder Site kann ein eigener Benutzer zugeordnet werden, mit dem der PHP-Prozess mit Windows kommuniziert (Bild 5)

PHP-Laufzeitumgebung

Ausserdem benötigt unser Projekt noch eine PHP-Laufzeitumgebung. Die PHP-Runtime wird bei Windows Azure mit dem Projekt veröffentlicht. Das hat den Vorteil, dass man volle Kontrolle über die Laufzeitumgebung hat. Wir kopieren die von windows.php.net/download/heruntergeladene Zip-Variante der gewünschten Version, in unserem Fall 5.3.1, und wählen dabei das Non-Thread-Safe VC9 x86-Derivat. Entzippt wandert die Runtime nun in unseren Projektordner und in unserem Fall in den Unterordner php-5.3.1. Auf diesen Ordner verweisen die beiden Web.-Konfigurationsdateien.

Einzig fehlt nun noch die php.ini-Datei. Diese liegt vorbereitet mit im Unterordner, allerdings mit der Erweiterung .ini-development beziehungsweise .ini-production. Sollen Änderungen an der php.ini vorgenommen werden, so empfiehlt es sich, vorerst die php.ini-development in php.ini umzubenennen und einen Testlauf durchzuführen. Den Nachweis, dass die php.ini-Datei geladen wird, liefert uns die Fehlerausgabe You are *required* to use the date.timezone setting [...], die uns inständig bittet, die korrekte Zeitzone in der php.ini zu setzen.

Wie bringe ich dieses Projekt nun in die Cloud? Auch dafür gibt es in Eclipse ein Kontextmenü. Nach einem Rechtsklick auf den Projektordner und Auswahl von Windows Azure lässt sich Publish Application to Windows Azure Portal selektieren. Daraufhin öffnet sich ein Datei-Browser und die Administrationsoberfläche von Windows Azure im Systembrowser. Man wählt den gewünschten Service aus, in den das Projekt veröffentlicht werden soll und lädt die Paketdatei und die ServiceConfiguration.cscfg aus dem geöffneten Datei-Browser hoch.

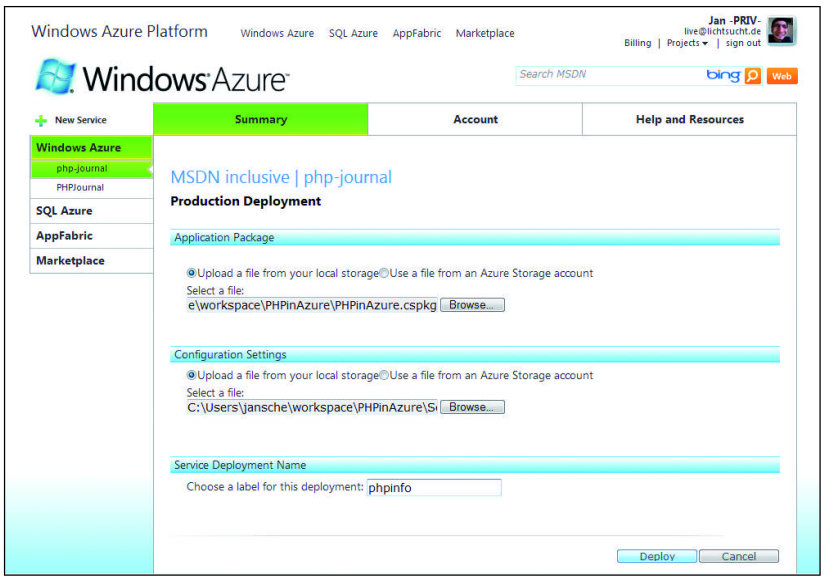
Nach einer kurzen Wartezeit, in der das Projekt provisioniert wird, steht die Web-Oberfläche wieder zur Verfügung. Nun lässt sich das Projekt mit dem Run-Button in den Livebetrieb bringen. Die Rollen, in unserem Fall nur eine Web-Rolle, werden initialisiert. Danach führt die auf der Seite angegebene URL zu unserem Web-Projekt (Bild 6).

Skalierung kann an zweierlei Orten stattfinden. Wir können bereits während der Entwicklung in der lokalen Runtime, der Development Fabric, Skalierung simulieren. Dazu öffnen wir

die Eigenschaften des Projektes mit Alt+Enter oder über Rechtsklick und Properties. Im Konfigurationsteil für Windows Azure befindet sich die Service Configuration, in der wir die Anzahl der Instanzen, die gestartet werden sollen, angeben können. Diese Änderung wird sowohl in der lokalen Simulation der Windows Azure Fabric abgebildet, als auch in den Livebetrieb übernommen. Im Livebetrieb gibt es aber eine flexiblere Methode, um ohne erneute Veröffentlichung die Instanzenanzahl zu verändern. Dazu wählt man den Button Configure..., woraufhin eine Textbox die Möglichkeit gibt die ServiceConfiguration.cscfg zur Laufzeit zu verändern.

Man muss sich gewisser Dinge bewusst sein, wenn man die Entscheidung für oder wider Cloud Computing treffen soll. So ist das Pay-As-You-Go-Modell eine wunderbare Sache, da für ungenutzte Ressourcen keine Kosten entstehen. Zum anderen aber muss man sich auch klar sein, dass diese quasi unendlichen Ressourcen, sollte man sie denn nutzen, Geld kosten. Natürlich hofft jeder Entwickler, jeder Erfinder und jeder Unternehmer, dass seine Idee erfolgreich ist. Dass sie den Hype erfährt, für den das Internet repräsentativ steht. Dass aus einer kleinen Idee ein rasant wachsender Publikumsmagnet wird, der es in die Top 10000 der beliebtesten Online-Applikationen schafft.

Allerdings sollte das Erlösmodell von Beginn an einen festen Platz in der Planung haben. Denn um die hoffentlich notwendigen Ressourcen bezahlen zu können reichen hohe Benutzerzahlen nicht aus. Wenn von heute auf morgen das einhundertfache an Rechenleistung oder Speicher benötigt wird, dann kostet das auch – Rabatteffekte beiseite gelassen – einhundertmal so viel. Um diese Kosten tragen zu können muss sich ein Anstieg der Benutzerzahlen einer öffentlich zugänglichen Web-Anwendung proportional auf die Erlöse auswirken. Für andere Anwendungszwecke von Windows Azure trifft diese Voraussetzung wiederum gar nicht zu, weil nur sicherer Speicherplatz für einen kurzen Zeitraum benötigt wird, in dem sich eigene Hardware nicht armortisieren würde. Microsofts Cloud Computing eröffnet hier neue Möglichkeiten. Oder es handelt sich um die Auslagerung einer bestehenden Intranet-Funktionalität, deren Benutzerkreis auf natürliche Art begrenzt ist. Es gibt zahlreiche Anwendungszwecke für Cloud Computing. Die Windows Azure Plattform bietet hinsichtlich Plattform und Infrastruktur zweifellos eine sorgenfreie Basis. Und wenn man einmal mit dem Komponentenmodell und dem Prinzip der Rollen warm geworden ist, dann ist es das vertrauenserweckendste und logischste Konzept, das Cloud Computing derzeit zu bieten hat. [mb]



mit dem Run-Button lässt sich das Projekt in den Livebetrieb bringen (Bild 6)

LISTING 3: WEB.CONFIG

```
<?xml version="1.0"?>
<configuration>
    <system.webServer>
      <handlers>
       <add name="FastGGI Handler"
            verb="*"
            path="*.php"
            scriptProcessor="%RoleRoot%\approot\php-5.3.1\php-cgi.exe|-c php.ini"
            modules="FastCgiModule"
            resourceType="Unspecified" />
      </handlers>
                <defaultDocument>
                <files>
                        <add value="index.php"/>
                </files>
                </defaultDocument>
        </system.webServer>
</configuration>
```