Doppelplusgut

Azure als Cloud-Plattform für Java-Anwendungen

Holger Sirtl

Microsofts Cloud-Plattform Azure steht seit Beginn des Jahres 2010 in einer ersten Version zum Einsatz bereit. Dabei ist Azure keineswegs auf rein Microsoft-basierte Infrastrukturen ausgelegt. Ein paar neue Funktionen machen die Plattform durchaus als Ausführungsumgebung für Java-basierte Anwendungen interessant. So lassen sich beispielsweise Java-basierte Serversysteme wie Apache Tomcat oder Apache Webserver auf Windows Azure betreiben. Dieser Artikel gibt einen Überblick über die Windows Azure Platform sowie die Möglichkeiten des Betriebs Java-basierter Anwendungssysteme auf Windows Azure.

Die Windows Azure Platform

Mit der Windows Azure Platform hat Microsoft seine Technologieplattform mit Beginn des Jahres 2010 um ein leistungsfähiges Cloud-Angebot erweitert. Die Plattform bietet drei Gruppen von Cloud-Services an, die einzeln oder in Kombination miteinander genutzt werden können: Windows Azure, SQL Azure und Azure AppFabric.

Windows Azure ist das Betriebssystem für Microsofts Cloud. Es stellt mit dem Windows Azure Compute Service eine Ausführungsumgebung für Cloud-Services zur Verfügung. Mit dessen Hilfe können selbstentwickelte Cloud-Anwendungen in Microsofts Cloud-Infrastruktur betrieben werden. Der Dienst bietet ein automatisiertes Management von Cloud-Services. Dazu gehören die Überwachung der ausgeführten Cloud-Services sowie die automatische Skalierbarkeit der Umgebung zur Anpassung der genutzten Umgebung an sich änderndes Lastverhalten.

Grundsätzlich lässt sich auf Windows Azure jede Software ausführen, die ohne Administratorrechte auch auf einem Windows-Server ausgeführt werden kann. Dies ist unabhängig von der Implementierungstechnologie. So können auch Java-Anwendungen auf Azure betrieben werden. Windows Azure stellt darüber hinaus Speicherdienste (Windows Azure Storage Services) zur Verfügung. In diesen können strukturierte Daten in Form von Entity Sets (Table Storage) und große Binärdaten (BLOBs) im sogenannten Blob Storage abgelegt werden.

Asynchroner Nachrichtenaustausch kann über Windows Azure Queues abgewickelt werden. Seit Neuestem stellt Windows Azure mit Azure Drives auch ein NTFS-basiertes Dateisystem zur persistenten Speicherung von Daten zur Verfügung. Die Speicherdienste können über Standardschnittstellen (SOAP, REST, XML), also von jeder Technologie, die diese Standards unterstützt – so natürlich auch Java –, genutzt werden.

Wer ein relationales Datenbanksystem in der Cloud benötigt, findet mit *SQL Azure* den passenden Dienst. SQL Azure unterstützt einen Teil der Funktionen eines SQL-Servers und stellt seine Funktionen über einen TDS-Endpunkt (TDS = Tabular Data Stream) zur Verfügung. Dieser ist kompatibel zu entsprechenden Endpunkten eines SQL-Servers, womit die Funktionen von SQL Azure über die gleichen Technologien und Werkzeuge genutzt werden können, die auch für einen SQL-Server zum Einsatz kommen. Aus Java heraus kann SQL Azure mithilfe eines entsprechenden JDBC-Treibers für SQL-Server genutzt werden.

Mit der Windows Azure Platform *AppFabric* stellt die Plattform auch einen Kommunikationsdienst bereit, über den verteilte Webservices, unabhängig von Implementierungstechnologie und Ausführungsort, über einen Service-Bus kommunizieren können. Dabei können die beteiligten Kommunikationspartner den Nachrichtenaustausch absichern und Teile der Authentifizierungs- und Autorisierungslogik an den Access Control Service auslagern, der Funktionen zur Claims-basierten Zugriffskontrolle bereitstellt.

Optionen zur Nutzung der Plattform

Wenngleich der Windows Azure Platform bewährte Windows-Server- und .NET-Technologie zugrunde liegt, ist ihre Nutzung keineswegs auf Microsoft-Technologien beschränkt. Alle Funktionen der Plattform sind über Webservices-Schnittstellen nutzbar. Für verschiedene Technologien (.NET, Java, PHP, Ruby) sind Software Development Kits (SDKs) kostenlos verfügbar. Diese erleichtern die Interaktion mit der Plattform. So kann zur Nutzung des Windows Azure Storage aus Java heraus eine Klasse `StorageClient` aus dem Windows Azure SDK für Java genutzt werden, um über einfache Methodenaufrufe Daten im Storage zu speichern, zu lesen usw.

Mithilfe der SDKs können zum einen Anwendungen erstellt werden, die auf Windows Azure ausgeführt werden, zum anderen auch Anwendungen, die außerhalb von Windows Azure betrieben werden (z. B. Rich Clients), die aber Funktionen der Windows Azure Platform nutzen (z. B. zu Speicherung von Daten im Windows Azure Storage, zur asynchronen Kommunikation über Azure Queues oder zur sicheren Vernetzung von Webservices über die AppFabric). Abbildung 1 skizziert diese Möglichkeiten.

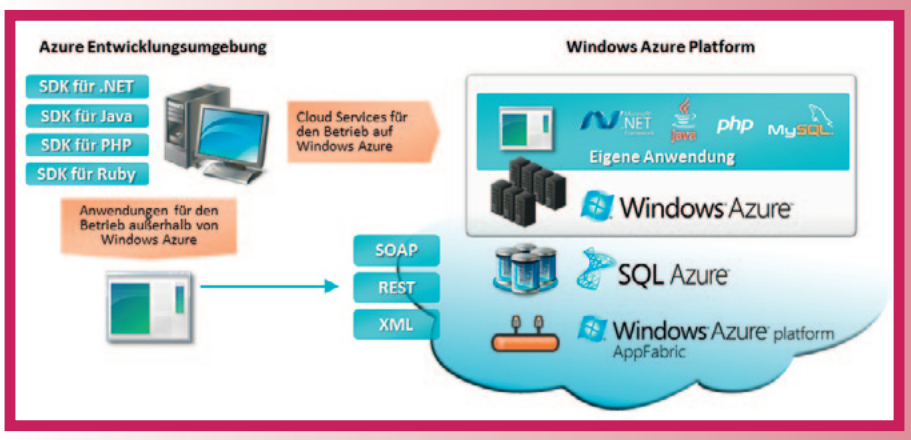

Abb. 1 Möglichkeiten der Nutzung der Windows Azure Platform

Zur Speicherung relationaler Daten in der Cloud über SQL Azure ist lediglich ein passender SQL-Server-Treiber erforderlich. Dieser kann auch für die Kommunikation mit SQL Azure

verwendet werden. Der Connection-String muss dabei auf eine SQL Azure-Instanz zeigen. Diese kann über das Portal unter `http://sql.azure.com` angelegt werden.

```
package de.azure.java;
import java.sql.*;
import com.microsoft.sqlserver.jdbc.SQLServerDataSource;

public class AzureSqlClient {
  public static void main(String[] args) {
    try {
      Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

      String connectionUrl =
        "jdbc:sqlserver://x0wa3wx4ue.database.windows.net;
        databaseName=TestDB;Username=user@x0wa3wx4ue;Password=****;";
      Connection con = DriverManager.getConnection(connectionUrl);

      String SQL = "SELECT Nachname, Vorname FROM [Kunden]";
      Statement stmt = con.createStatement();
      ResultSet rs = stmt.executeQuery(SQL);

      while (rs.next()) {
        System.out.println(rs.getString("Nachname") + ", " +
          rs.getString("Vorname");
        }
        rs.close();
        stmt.close();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

Listing 1: Zugriff auf SQL Azure aus Java heraus

Architektur von Cloud-Services für Azure

Anwendungen, die auf Windows Azure ausgeführt werden sollen, müssen ein paar formalen Kriterien entsprechen. Grundsätzlich müssen diese Anwendungen in Services untergliedert werden, die bestimmte Schnittstellen implementieren. Windows Azure unterscheidet dabei zwischen zwei Arten von Services:

- *Web Roles* sind Services, die später auf Windows Azure in einer Virtuellen Maschine (VM) innerhalb eines IIS (Microsoft Internet Information Services) gehostet werden. Web Roles können Nachrichten über die üblichen http-Ports empfangen und stellen typischerweise Web-Frontends bzw. Webservices zur Verfügung. Windows Azure übernimmt die Konfiguration des IIS sowie des vorgeschalteten Lastverteilers.
- *Worker Roles*, die zweite Art von Cloud-Services für Azure, werden auf Windows Azure von einem Systemprozess gehostet. Ursprünglich konnten diese Services keine Nachrichten von außerhalb Azures empfangen und waren ausschließlich für die Abarbeitung von Hintergrundprozessen vorgesehen. Inzwischen ist es aber möglich, auch für Worker Roles Endpunkte und entsprechende Ports selbst zu konfigurieren, um über diese Ports Nachrichten von außen zu empfangen. Die Konfiguration muss hierbei aber vom Entwickler des Cloud-Service durchgeführt werden. Worker Roles werden grundsätzlich als Endlosschleifen implementiert.

Anwendungen, die auf Windows Azure ausgeführt werden sollen, müssen also verschiedene Vorgaben einhalten. Im Rahmen der Migration einer Anwendung nach Azure fallen deshalb gewisse Aufwände für die Anpassung an. Als Gegenleistung bietet Windows Azure dann ein automatisiertes Management der Anwendung. Auf Knopfdruck können beispielsweise Instanzen von Web oder Worker Roles hinzugeschaltet werden. Windows Azure kümmert sich um die korrekte Konfiguration der VMs, die Anpassung der Lastverteilerkonfiguration, die Überwachung der laufenden Anwendungen usw.

Ausführung Java-basierter Anwendungen auf Azure

Die VMs von Windows Azure enthalten unter anderem eine spezielle Windows-Server-Version mit vorinstallierter .NET Framework 3.5 SP1 Runtime (bzw. im Falle von Web Roles noch einen vorinstallierten Microsoft Internet Information Services 7). .NET-basierte Cloud-Services sind somit unmittelbar ausführbar. Sollen Java-Services ausgeführt werden, so ist es erforderlich, dem Service-Paket, welches auf Windows Azure installiert wird, eine Java-Laufzeitumgebung (JRE) hinzuzufügen. Somit kann jeder Java-Anwendung das passende JRE mitgegeben werden. Die Java-Klasse muss von einer kleinen .NET-Wrapper-Klasse aus in einem neuen Prozess gestartet werden.

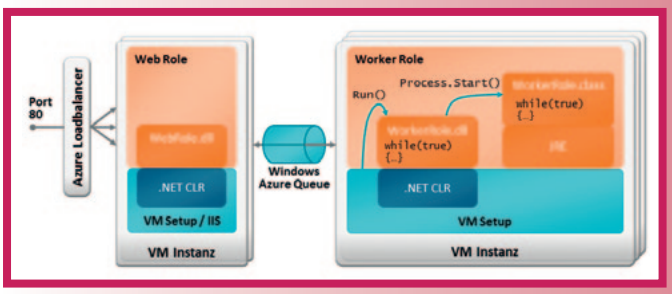

Abb. 2: Ausführung einer Java-Klasse auf Windows Azure

Abbildung 2 zeigt den typischen Aufbau einer Anwendung, in der in einer Worker Role ein Java-Prozess ausgeführt wird. Die Web Role implementiert das Frontend und sendet Arbeitsaufträge über eine Queue an die Worker Role. Bei der Installation der Anwendung initialisiert Windows Azure die Worker Role durch einen Aufruf der `Run()`-Methode der `WorkerRole`-Klasse. In dieser Methode kann entsprechend Listing 2 ein neuer Prozess zur Ausführung der Java-Klasse gestartet werden.

```
public override void Run() {
  StreamReader sr;
  string returnDetails;

  Process p = new Process() {
    StartInfo = new ProcessStartInfo(Path.Combine(
      Environment.GetEnvironmentVariable("RoleRoot"),
      @"jre\bin\java.exe"), @"-cp lib;lib\*;. worker") {
      RedirectStandardInput = true,
      RedirectStandardOutput = true,
      RedirectStandardError = true,
      UseShellExecute = false,
    }
  };
  p.EnableRaisingEvents = false;
  p.Start();
  while (true) {
    Thread.Sleep(10000);
    Trace.WriteLine("Working", "Information");
  }
}
```

Listing 2: Aufruf einer Java-Klasse aus der Initialisierungsmethode einer Worker Role

Der in Abbildung 2 skizzierte Aufbau der Anwendung ist erforderlich, wenn eingehende Requests zunächst von einer

Web Role verarbeitet werden, die wiederum auf einem IIS ausgeführt wird. Es ist allerdings auch möglich, einen Java-basierten Anwendungsserver wie Tomcat auf Windows Azure zu betreiben.

Apache Tomcat auf Azure

Da Tomcat als Teil einer Worker Role in diesem Fall die Rolle des http-Servers übernehmen soll, muss die Konfiguration der Ports durch den Entwickler übernommen werden. In der Konfigurationsdatei der Worker Role wird hierzu ein Endpunkt (z. B. Port 80) definiert. Eine entsprechende Konfiguration ist in Listing 3 zu sehen. Dort wird Tomcat noch Speicher im lokalen Dateisystem zur Ablage von Logdateien usw. bereitgestellt.

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="TomcatRole" ...>
  <WorkerRole name="TomcatWorkerRole" enableNativeCodeExecution="true">
    ...
    <Endpoints>
      <InputEndpoint name="Tomcat" port="80" protocol="tcp" />
    </Endpoints>
    <LocalResources>
      <LocalStorage cleanOnRoleRecycle="false"
                    name="TomcatLocation" sizeInMB="2048" />
    </LocalResources>
  </WorkerRole>
</ServiceDefinition>
```

Listing 3: Konfiguration der Tomcat-Endpunkte in der Worker Role

Der Lastverteiler kann nun Requests über Port 80 empfangen und leitet diese an die verschiedenen Instanzen der Worker Role weiter, von denen jede auf einem anderen internen Port empfängt. Bei der Initialisierung der Worker Role muss deshalb in der Datei „server.xml“ der korrekte (interne) Empfangs-Port konfiguriert werden. Dies kann wieder in der Initialisierungsmethode `Run()` geschehen, in der zur Laufzeit der interne Port ermittelt werden kann (z. B. Port 5100). Danach müssen noch die zur Ausführung von Tomcat erforderlichen Umgebungsvariablen gesetzt werden, ehe letztlich Tomcat in einem neuen Prozess gestartet wird. Der Entwickler muss somit neben der .NET-basierten Worker Role-Basisklasse die Java-Runtime und Tomcat in die Worker Role packen und das Gesamtpaket auf Windows Azure installieren.

Das Gesamtsystem dieses Szenarios ist in Abbildung 3 gezeigt. Mit Hilfe des kostenlos über die MSDN Code Gallery erhältlichen Tomcat Solution Accelerators [AzureTomcat] werden diese Konfigurationsschritte deutlich vereinfacht.

Wenngleich zur Installation von Tomcat einige zusätzliche Arbeitsschritte erforderlich sind, lohnt sich der Aufwand. Ausgeführt auf Windows Azure profitiert auch Tomcat von dem automatisierten Umgebungsmanagement der Plattform. So werden beispielsweise die Instanzen überwacht und bei Bedarf neu gestartet oder Betriebssystem-Patches ohne Zutun des Entwicklers eingespielt. Auf Knopfdruck können zusätzliche Instanzen einer Worker Role (samt JRE und Tomcat) gestartet und über den Lastverteiler aufgerufen werden.

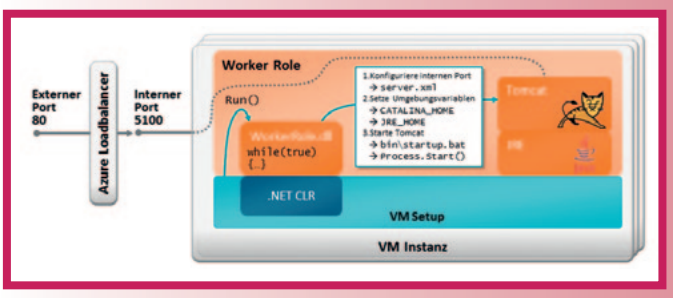

Abb. 3: Ausführung von Tomcat in einer Worker Role auf Windows Azure

Fazit

Die Windows Azure Platform stellt nicht nur .NET-Entwicklern, sondern insbesondere auch Java-Entwicklern eine Plattform zur Nutzung und Ausführung von Cloud-Services zur Verfügung. Aus Java-Anwendungen können mithilfe eines entsprechenden SDKs Azure Services leicht genutzt werden. Es besteht aber auch die Möglichkeit, Java-basierte Anwendungen bzw. Anwendungsserver wie Tomcat auf Azure zu betreiben. Hierbei kann der Entwickler die passende Umgebung selbst wählen und konfigurieren. Damit profitieren auch Java-Anwendungen von dem automatisierten Anwendungs- und Umgebungsmanagement von Azure.

Links

[Azure] Windows® Azure™ Platform, `http://www.azure.com/`
[AzureInterop] Windows Azure Interoperabilität, `http://www.microsoft.com/WindowsAzure/interop/`
[AzureTomcat] Tomcat Solution Accelerator, `http://code.msdn.microsoft.com/winazuretomcat/Release/ProjectReleases.aspx?ReleaseId=3550`
[Sirtl09] H. Sirtl, Microsoft Azure für Java-Entwickler, in: JavaSPEKTRUM, 6/2009

Holger Sirtl arbeitet bei der Developer Platform & Strategy Group der Microsoft Deutschland GmbH.
E-Mail: holger.sirtl@microsoft.com.