Über den Wolken

Microsoft Azure für Java-Entwickler

Holger Sirtl

Die Windows Azure Platform hat für Java-Entwickler einiges zu bieten. Windows Azure kann als hoch-verfügbare, flexibel skalierbare Cloud-Laufzeitumgebung für Java-Prozesse genutzt werden. Auf Daten in Windows Azure Storage und SQL Azure kann über bekannte Schnittstellen aus Java-Anwendungen heraus zugegriffen werden. Mit der Windows Azure AppFabric kann zudem der Brückenschlag zwischen der Java- und der .NET-Welt umgesetzt werden. Dieser Artikel gibt einen Überblick über die Azure Services und stellt Werkzeuge und Klassenbibliotheken vor, mit denen Java-Entwickler Azure effizient nutzen können.

Cloud Computing mit Azure

Die Windows Azure Platform ist Microsofts Platform-as-a-Service(PaaS)-Angebot, welches Ende 2009 im Markt eingeführt und von da an kontinuierlich weiterentwickelt wurde. Diese Cloud-Plattform richtet sich speziell an Softwareentwickler, die hoch-skalierbare, hoch-verfügbare, nutzungsabhängige Software-Dienste benötigen, um auf deren Basis leistungsfähige Cloud-Dienste zu implementieren.

Die Plattform ist aus drei Dienstgruppen aufgebaut, die einzeln oder in Kombination genutzt werden können:

- *Windows Azure* ist das Herzstück der Plattform und Microsofts Cloud-Betriebssystem. Es stellt eine automatisch verwaltete Laufzeitumgebung für Cloud-Dienste sowie Non-SQL-Speicherdienste, die über REST-Schnittstellen genutzt werden können, zur Verfügung. Microsoft hält die Plattform stets auf dem aktuellsten Stand, d. h. Upgrades und Patches werden auf Wunsch ohne Zutun des Nutzers eingespielt, wobei ein unterbrechungsfreier Betrieb der Dienste gewährleistet wird. Dies senkt die Administrationsaufwände für die Nutzer auf ein Minimum.
- *SQL Azure* ist ein SQL-Server-kompatibles, relationales Datenbankmanagementsystem (RDBMS), das als Cloud-Dienst bereitgestellt wird. Die Schnittstellenkompatibilität zu SQL-Server stellt sicher, dass bekannte Entwicklungs- und Managementwerkzeuge sowie Aufruftechnologien weiterverwendet werden können.
- Die *Windows Azure AppFabric* ist Kommunikations- und Integrationsplattform für Webservices, die in der Cloud oder außerhalb betrieben werden. Mithilfe der AppFabric kann ein sicherer Brückenschlag von der Cloud in klassische Unternehmens-IT über Unternehmens- und Technologiegrenzen hinweg realisiert werden.

Die Dienste der Plattform sind über REST-basierte Schnittstellen zugänglich, die Nutzung setzt also nicht voraus, dass ausschließlich Microsoft-Technologien wie .NET eingesetzt werden. Insbesondere Java-Entwickler haben die Möglichkeit, Java-Prozesse auf Windows Azure zu betreiben, Cloud-basierte Speicherdienste aus Windows Azure bzw. SQL Azure zu nutzen und mittels Windows Azure AppFabric Kommunikation (auch mit Nicht-Java-Prozessen) sicher über die Cloud abzuwickeln.

Laufzeitumgebung für Java-Prozesse

Anwendungen, die auf Windows Azure ausgeführt werden, sind aus Diensten, sogenannten Rollen, aufgebaut, deren Instanzen jeweils in eigenen Virtuellen Maschinen (VMs) ausgeführt werden. Eine Anwendung muss mindestens eine Instanz einer Rolle, d. h. in der Ausführung mindestens eine VM, enthalten. Der Zugriff auf eine Azure-basierte Cloud-Anwendung erfolgt grundsätzlich über einen Loadbalancer, der die Requests an eine (zufällig ausgewählte) Instanz der aufgerufenen Rolle weiterleitet.

Es gibt drei Arten von Rollen, die in Abbildung 1 skizziert sind. Sie unterscheiden sich jeweils darin, welche Komponenten der VM von Windows Azure und welche vom Entwickler bereitgestellt werden. In *Web Roles* werden von Windows Azure das Gast-Betriebssystem (z. B. Windows Server 2008 R2), ein Management-Layer (zu automatischer Überwachung und Management der VM) sowie eine Installation des Internet Information Servers (IIS) bereitgestellt. Ein Entwickler stellt nur die eigentliche Anwendungslogik bereit, die im IIS gehostet wird. Web Roles eignen sich demnach für Webanwendungen, die in .NET, PHP oder anderen über IIS verwaltbaren Technologien entwickelt wurden. Aus diesen heraus können Java-Anwendungen als neue Prozesse gestartet werden.

Interessanter sind für Java-Entwickler allerdings die beiden anderen Rollen. *Worker Roles* unterscheiden sich von Web Roles nur dadurch, dass kein IIS als Webserver vorinstalliert ist. Für Java-Entwickler bietet sich nun die Möglichkeit, eine eigene JVM und bei Bedarf einen eigenen Webserver (z. B. Apache Webserver) oder Servlet-Container (z. B. Tomcat) bereitzustellen und Java-Prozesse darin auszuführen. Benötigt wird lediglich ein kleiner .NET-Bootstrapper, der die entsprechenden Java-Prozesse startet. Mit dem Windows Azure Tomcat Solution Accelerator [ATSA] stellt Microsoft Werkzeuge bereit, mit denen ein entsprechendes Anwendungspaket (mit einer frei wählbaren JVM- und Tomcat-Distribution) einer Worker Role mit wenigen Schritten erzeugt werden kann.

Bei der dritten Rollenart, den *VM Roles*, hat der Entwickler noch größere Flexibilität: Er kann ein vollständiges, auf Windows Server 2008 R2 basierendes VM-Image erstellen und auf Windows Azure betreiben. Hier stehen also alle Möglichkeiten zur Vorkonfiguration der VM, einschließlich eigener

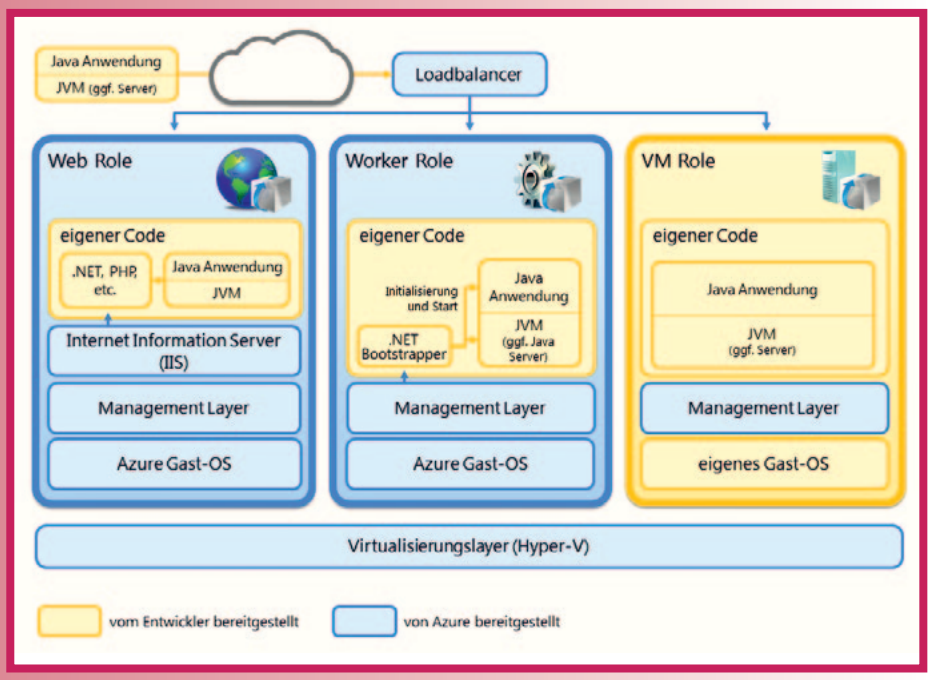

Abb. 1: Nutzung von Windows Azure als Laufzeitumgebung für Java-Prozesse

Anwendungsserver, zur Verfügung. Diese Flexibilität wird allerdings mit Einschränkungen beim Management der VM erkauft. Während bei Web und Worker Roles Windows Azure die Rollen überwacht und bei Bedarf neu startet und das Gast-Betriebssystem automatisch mit den jeweils neuesten Patches und Upgrades bestückt, ist bei VM Roles der Entwickler selbst für die Administration des Gast-Betriebssystems verantwortlich.

Für jede Rolle kann unabhängig bestimmt werden, in wie vielen Instanzen sie ausgeführt werden soll. Diese Instanzenanzahl kann zur Laufzeit geändert werden, die Zahl kann also an die anliegende Last angepasst werden. Dieser Prozess ist automatisierbar. Windows Azure stellt jeweils sicher, dass immer die spezifizierte Anzahl an Rolleninstanzen ausgeführt wird, d. h. vollautomatisch werden zusätzliche Instanzen gestartet oder heruntergefahren bzw. ausgefallene Instanzen ersetzt.

Datenspeicherung in der Cloud

Die Windows Azure Platform stellt sowohl SQL- als auch Non-SQL-Datenspeicher zur Verfügung. Unabhängig vom Speichertyp werden Daten in Azure grundsätzlich dreifach redundant gespeichert. Während die SQL-Funktionalitäten von SQL Azure abgedeckt werden, sind die Non-SQL-Datenspeicher Windows Azure zugeordnet. Abbildung 2 zeigt die alternativen Speichermöglichkeiten, die alle aus Java heraus genutzt werden können.

Windows Azure stellt drei Arten von Non-SQL-Speichern zur Verfügung. Im *Table Storage* können strukturierte Daten in großer Menge abgelegt werden. Tabellen sind dort allerdings keine Tabellen im relationalen Sinne, sondern Entity Sets, deren Einträge nur einen Primärschlüssel besitzen. Ein Entity ist dabei eine Menge von typisierten Propertys.

Queue Storage stellt Warteschlangen-Funktionalität zur Verfügung. Kommunikationspartner können hierüber Nachrichten austauschen, die im „best effort first in, first out"-Prinzip zugestellt werden. Das Auslesen einer Nachricht aus einer Queue führt zunächst dazu, dass die Nachricht für einen vom Entwickler festgelegten Zeitraum unsichtbar wird. Wird sie innerhalb dieses Zeitraums nicht vom Empfänger (z. B. nach erfolgreicher Verarbeitung) gelöscht, wird sie wieder sichtbar und kann von weiteren Empfängern gelesen werden. Dies stellt sicher, dass Nachrichten nicht aufgrund des Ausfalls eines Empfängers verloren gehen.

Blob Storage kann dazu verwendet werden, große Binärdaten (BLOB = binary large object) wie z. B. Videodateien in der Cloud zu speichern. Blobs können über Container hierarchisch verschachtelt abgelegt werden.

Die Speicherungsdienste von Windows Azure können über eine REST-Schnittstelle angesprochen werden. Aus Java können entsprechende REST-Aufrufe über simple HTTP-Requests implementiert werden. Eine einfachere Möglichkeit ergibt sich aus der Verwendung des Windows Azure SDK für Java [ASDKJD]. Dieses stellt eine Klassenbibliothek bereit, mit deren Hilfe Windows Azure Storage sehr leicht und intuitiv angesprochen werden kann.

Listing 1 zeigt ein Code-Fragment, in dem mithilfe des SDK und dessen Klassen zunächst alle verfügbaren Tabellen aus dem Table Storage aufgelistet und anschließend alle Entitys der Tabelle `WADPerformanceCountersTable` aufgelistet werden. Auf umfangreiche Fehlerbehandlung wurde aus Gründen der Übersichtlichkeit verzichtet.

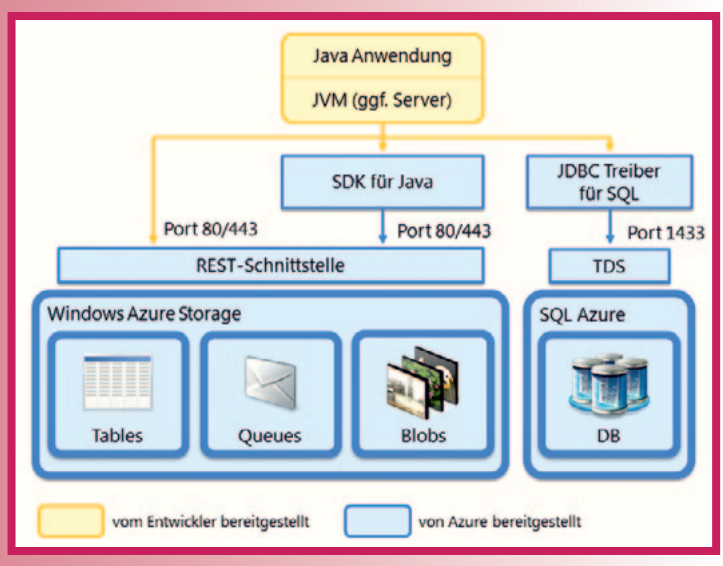

Abb. 2 Alternativen für die Datenspeicherung in Azure

```
public static void main(String[] args) {
  try {
    TableStorageClient tableClient =
      createStorageAccess(AZURE_ACCOUNT_NAME, AZURE_ACCOUNT_KEY);

    // Liste alle Tabellen auf
    List<String> tables = tableClient.listTables();
```

```
    if (!tables.isEmpty()) {
      System.out.println("List of Tables in Table Storage:");
      for (String tableName : tables) {
        System.out.println(tableName);
      }
    } else {
      System.out.println("No Tables in Table Storage\n");
    }
    // Liste alle Entitys aus der Tabelle
    // WADPerformanceCountersTable auf
    ITable table = tableClient.getTableReference(
                    "WADPerformanceCountersTable");
    List<ITableServiceEntity> tableEntities =
      new ArrayList<ITableServiceEntity>();
    queryEntitiesByKeys(table, tableEntities, null, null);
    if (!tableEntities.isEmpty()) {
      for (ITableServiceEntity tableEntity : tableEntities) {
        System.out.printf("PartitionKey '%s'\tRowKey '%s'\t",
          tableEntity.getPartitionKey(), tableEntity.getRowKey());
        for (ICloudTableColumn tableColumn : tableEntity.getValues()) {
          System.out.printf("(%s)'%s'\t",
            tableColumn.getType().getLiteral(),
            tableColumn.getValue());
        }
        System.out.printf( "\n" );
      }
    }
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

Listing 1: Zugriff auf Windows Azure Table Storage mithilfe des Windows Azure SDK für Java

SQL Azure stellt ein echtes relationales Datenbankmanagementsystem als Cloud-Dienst bereit. Enthalten sind ein Subset der Datenbank-Engine aus SQL-Server, die SQL Reporting Services sowie der SQL Sync Service, mit dem SQL Azure-Datenbanken mit lokalen Datenbanken synchronisiert werden können. Für die Migration von einem bestehenden lokalen SQL-Server nach SQL Azure wird von Microsoft der SQL Azure Migration Wizard [ASQLMW] bereitgestellt, mit dem das Datenbankschema und die Daten selbst nach SQL Azure übertragen werden können.

Für den Zugriff auf SQL Azure aus Java-Anwendungen heraus wird lediglich ein JDBC-Treiber benötigt. Aufgrund der Schnittstellenkompatibilität kann ein bestehender Treiber für SQL-Server verwendet werden. Entsprechend gestaltet sich auch die Java-Aufruflogik analog dem Zugriff auf einen SQL-Server wie in Listing 2 zu sehen ist. Dort werden aus der Kunden-Tabelle einer SQL Azure-Datenbank alle Namen ausgelesen und angezeigt.

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

    SQLServerDataSource ds = new SQLServerDataSource();
    ds.setUser(Resource.username);
    ds.setPassword(Resource.password);
    ds.setServerName("p4rzelagse.database.windows.net");
    ds.setPortNumber(1433);
    ds.setDatabaseName("JavaDemoDB");
    Connection con = ds.getConnection();
    String SQL = "SELECT Nachname FROM [Kunden]";
    Statement stmt = con.createStatement();
    ResultSet rs = stmt.executeQuery(SQL);
    while (rs.next()) {
      System.out.println(rs.getString("Nachname"));
    }
    rs.close();
    stmt.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

Listing 2: Zugriff auf eine SQL Azure-Datenbank über SQL-Server-JDBC-Treiber

Integration von Java- und .NET-Services

Die sichere Integration heterogener, verteilter Anwendungskomponenten über Firewall- und Unternehmensgrenzen hinweg ist für Entwickler immer eine Herausforderung. Um einen derartigen Kommunikationsmechanismus sicher zu implementieren, kann die Windows Azure AppFabric als Vermittlerkomponente eingesetzt werden. Diese kann dabei helfen, den Kommunikationsfluss über einen Service-Bus zu etablieren und Teile der Authentifizierung der Kommunikationspartner via Access Control Service zu übernehmen.

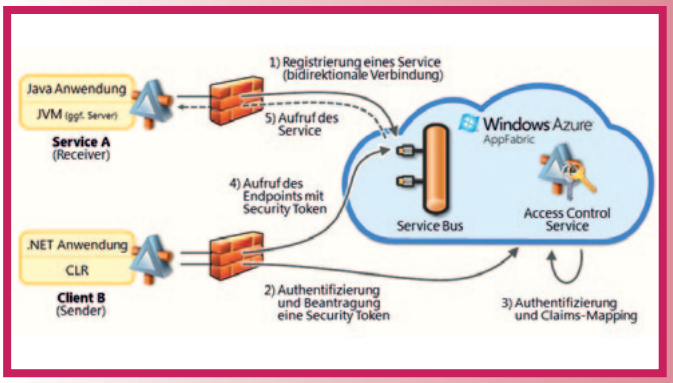

Abb. 3: Integration heterogener, verteilter Services über Firewall- und Unternehmensgrenzen hinweg

Abbildung 3 zeigt, wie die Kommunikation über die Windows Azure AppFabric abgewickelt werden kann. In dem abgebildeten Szenario soll es ermöglicht werden, dass der in .NET implementierte Client B über die Firewalls hinweg mit dem in Java implementierten Service A kommunizieren kann.

1. Im ersten Schritt registriert sich A beim Service-Bus und teilt diesem bei der Registrierung mit, welche Voraussetzungen ein Kommunikationspartner erfüllen muss, damit dieser ihn aufrufen darf. Im Rahmen der Registrierung baut A eine bidirektionale Kommunikationsverbindung mit dem Service-Bus auf, d. h. über diese Verbindung kann A Nachrichten senden und empfangen. Ebenso kann er definieren, über welche Authentifizierungsmechanismen (z. B. Windows Live ID) sich der Partner authentifizieren muss. Der Service-Bus teilt A dann einen Kommunikationsendpunkt, d. h. eine bestimmte Internet-Adresse innerhalb eines festgelegten Namensraumes, mit, über den dieser dann von der Außenwelt angesprochen werden kann.
2. Client B kann sich dann beim Access Control Service (ACS) anmelden und ein für den Aufruf von A benötigtes Security-Token beantragen. Der ACS authentifiziert B, d. h. ruft den passenden Identity Provider (z. B. Windows Live ID) auf, um sich die Daten bzw. Claims von B zu beschaffen.

❸ Diese Claims bildet der ACS auf die entsprechenden Felder des von A angeforderten Security-Tokens ab und liefert diesen Token an B zurück.
❹ B kann diesen Token beim Aufruf des Endpunkts von A auf dem Service-Bus übergeben.
❺ Der Service-Bus leitet diesen Aufruf einschließlich des Security-Tokens über die zuvor von A etablierte Kommunikationsverbindung an A weiter. A kann durch Auswertung des Security-Tokens bestimmen, ob es den Zugriff auf die angeforderten Ressourcen zulässt.

Über die AppFabric ist es also möglich, über Firewall-Grenzen hinweg sichere Kommunikationsbeziehungen zwischen verteilten Systemen zu etablieren. Der Mechanismus ist dabei unabhängig vom Ausführungsort und der Implementierungstechnologie der beteiligten Kommunikationspartner (.NET, Java usw.). Der Access Control Service ermöglicht es zudem, Teile der Authentifizierung auszulagern. Die Implementierung eines empfangenden Service kann somit von aufwendiger Authentifizierungslogik entlastet werden. Für Java-Entwickler gibt es ein AppFabric SDK [AppFabric], mit dessen Hilfe auf Java-Seite entsprechende Kommunikationslogik implementiert werden kann.

Fazit

Die Windows Azure Platform stellt mit Windows Azure, SQL Azure und der Windows Azure AppFabric Plattformdienste zur Verfügung, mit deren Hilfe Java-Entwickler Cloud-Dienste entwickeln und in der Cloud betreiben, Daten effizient in der Cloud speichern und Kommunikation sicher über die Cloud abwickeln können.

Links

[AppFabric] AppFabric SDK for Java Developers,
`http://www.jdotnetservices.com/`
[ASDKJD] Windows Azure SDK for Java Developers,
`http://www.windowsazure4j.org/`
[ASQLMW] SQL Azure Migration Wizard,
`http://sqlazuremw.codeplex.com/`
[ATSA] Windows Azure Tomcat Solution Accelerator,
`http://code.msdn.microsoft.com/winazuretomcat`
[Azure] Deutsche Einstiegsseite für Azure,
`http://www.microsoft.de/azure`

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