Entwicklung in der Cloud mit Platform as a Service

Eine Idee setzt sich durch

Cloud Computing: Seit den ersten Anfängen entwickelte sich aus der Idee, Software als Dienst anzubieten, eine neue Philosophie in der IT-Welt, die sowohl Entwickler von Plattformanwendungen als auch Benutzer gleichermaßen begeistert. Zeit also, einen Blick auf aktuelle Ereignisse und auch die derzeitige Technik dahinter zu werfen.

von Andreas Holubek und Claudia Matthias

Software as a Service kennzeichnet eine Form der Verteilung und Nutzung von Software, die direkt auf das moderne Internet zugeschnitten ist. Einfach ausgedrückt wird hierbei eine Anwendung vom Anbieter gemietet und über Web Access genutzt. Tatsächlich aber geht Software as a Service über das reine Mieten von Software hinaus, denn neben der eigentlichen Anwendung werden auch alle begleitenden Dienste zur Verfügung gestellt. Dazu gehören der erforderliche Speicherplatz (Storage as a Service), Wartung und Updates sowie die benötigten Werkzeuge, um die Software an die jeweiligen Bedürfnisse des Anwenders anzupassen. Mit der heute verfügbaren Bandbreite und Anschlussqualität sind solche On-Demand-Lösungen populär geworden. Blickt man etwas tiefer in das Architekturprinzip einer Software-as-a-Service-Lösung, werden weitere, wichtige Details sichtbar. So nutzen alle Anwender einen gemeinsamen Kern, sind also als Mandanten zu verstehen. Im Gegensatz zum Application Service Providing (ASP), bei dem für jeden Benutzer eine eigene Instanz der Anwendung existiert, lassen sich somit gemeinsam genutzte Ressourcen über verschiedenste Benutzer verteilen (Abb. 1). Dieses der SaaS-Architektur eigene Konzept wird als Multi-Tenancy bezeichnet.

Ein weiteres, wichtiges Merkmal von Software as a Service besteht darin, dass die gemieteten Anwendungen in weiten Bereichen anpassbar sind, dabei jedoch immer in ihrer Domain ihres Bereichs bleiben. Ein sehr gutes Beispiel hierfür ist die CRM-Lösung von salesforce.com [1, 2]: hochgradig konfigurierbar, mit Entwicklungsumgebung für Workflows, benutzerdefinierte Businesslogik und zahlreichen weiteren Merkmalen.

Platform as a Service

Folgt man dem Gedanken, dass Geschäftsanwendungen wie CRM häufig den gleichen Grundaufbau haben, kommt man von Software as a Service zu Platform as a Service (PaaS). Der Kern einer Anwendung sowie ihre Architektur werden durch die Plattform vorgegeben. Platform as a Service bedeutet, dass alle benötigten Artefakte für die Entwicklung einer Soft-

ware as a Service vorhanden sind – von der Datenbank bis hin zum integrierten Entwicklungswerkzeug. Auch um komplexe Dinge wie die Skalierbarkeit der Anwendung und Hardware oder das Bereitstellen einer integrierten Laufzeitumgebung muss sich der Entwickler nicht mehr kümmern. Technisch interessant ist, dass die so erstellten Anwendungen auf einem gemeinsamen Kern laufen. Somit können einmal erstellte Tabellen, Businessobjekte, Web Services, Anwendungsteile u. v. m. von beliebiger Seite nachgenutzt werden. Die Vorteile für Entwickler und spätere Anwender liegen auf der Hand. Nachstehend seien nur einige genannt:

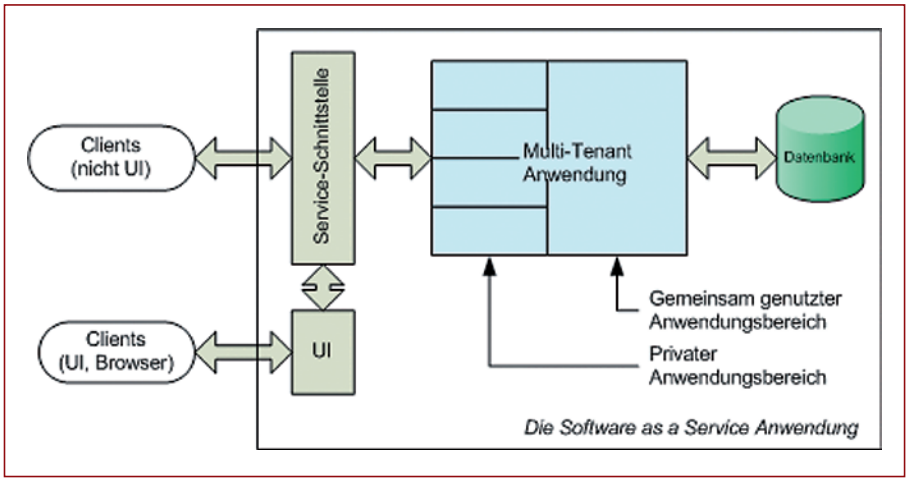

Abb. 1: Software-as-a-Service-Systemarchitektur

- Basis der auf Platform as a Service entwickelten Anwendungen ist eine Multi-Tenant-Architektur.
- Die Rechenleistung wird dynamisch verteilt, je nach Anforderung der Benutzer.
- Nur jene Ressourcen werden zur Verfügung gestellt, die tatsächlich benötigt werden – sie können somit effizienter und umweltschonender eingesetzt werden.
- Der Anbieter stellt eine vollständige Entwicklungsumgebung und Infrastruktur, Datenbank sowie Speicherplatz zur Verfügung (Infrastructure as a Service, Database as a Service und Storage as a Service).
- Internationale und nationale Sicherheitsrichtlinien werden beachtet und sind wie ISO-27001-Zertifizierung im Kern vorhanden.
- Metadaten und metadatengetriebene Entwicklung können genutzt werden.

Cloud Computing

Im Jahr 2008 setzte ein wahrer Boom im Bereich webbasierter IT-Nutzung ein. Für die Gesamtheit dieser „As a Service"-Konzepte setzte sich in der Industrie der Begriff „Cloud Computing" durch. Die verschiedenen Angebote – als Einzelservices oder Bündel von Dienstleistungen genutzt – umfassen dabei neben Software as a Service und Platform as a Service noch Konzepte wie Storage as a Service, User Interface as a Service, Logic as a Service, Integration as a Service oder Database as a Service. Immer gilt: Alle diese Ressourcen werden bei Bedarf gemietet und auch nutzungsbasiert bezahlt. Es ist ein Trend erkennbar, dass Firmen ihre IT-Infrastruktur zunehmend in Richtung der Cloud verschieben. Gerade die Markterfolge der Software-as-a-Service- und Platform-as-a-Service-Anbieter machen das sehr deutlich. Natürlich sollte man auch einige Besonderheiten nicht aus den Augen

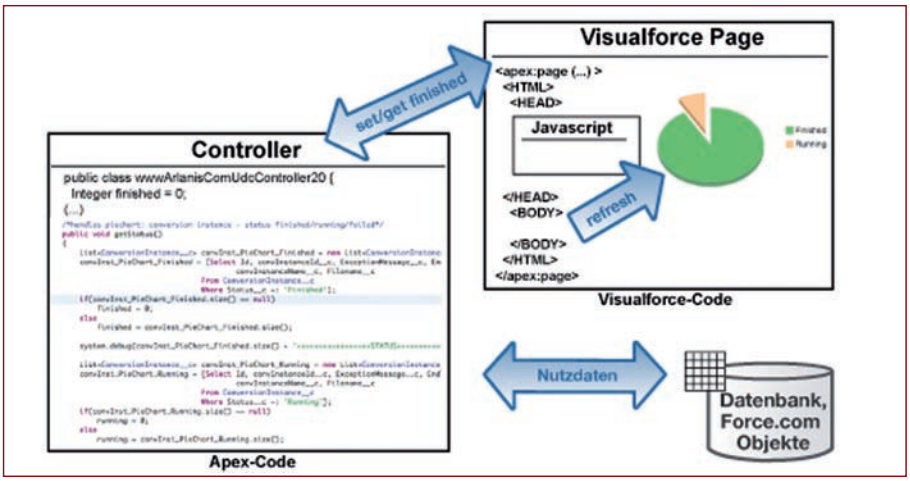

Abb. 2: Visualisierung von externen Informationen als Cloud-Anwendung

verlieren: So ist in diesem Umfeld derzeit noch keine echte Standardisierung erkennbar. Manche Unternehmen scheuen noch davor zurück, ihre Daten außer Haus zu geben, und neben höchstmöglichen Sicherheitsstandards müssen die Cloud-Computing-Anbieter auch eine 99,xx %ige Erreichbarkeit ihrer Sevices garantieren. Die derzeitig existierenden Plattformen und Anbieter zeigen jedoch, dass mit den Besonderheiten des Cloud Computings gut umgegangen werden kann.

Praxisbeispiel: Anwendungsentwicklung mit Force.com

Am Anfang jeder webbasierten Programmierung steht die Entscheidung für eine Plattform. Neben der *Custom Cloud Force.com* von salesforce.com, existieren heute bereits weitere, webbasierte Plattformen in verschiedenen Ausbauformen und mit Funktionalitäten für eine Vielzahl von Businessmodellen. Das nachstehende Beispiel (ein kleiner Teil einer realen Anwendung aus der Firma der Autoren) basiert auf Force.com [3]. Die entstehende Anwendung wird direkt im *salesforce.com*-Bereich abgelegt (neben dem Anwendungs-Tab sind deshalb auch noch die Sales Tabs zu sehen). Damit wird eine so genannte native Anwendung erstellt. Eine Anwendung also, die ohne Installation auf Seiten des Benutzers auskommt. Native Anwendungen werden ausschließlich mithilfe der Metadaten und dem Force.com IDE entwickelt. Sie benutzen weder das interne API direkt noch haben sie Referenzen oder Links auf Anwendungen außerhalb der Plattform. Die Kommunikation mit der restlichen Welt über Web Services ist jedoch möglich. Neben dieser Form der Anwendung gibt es noch Clientanwendungen. Diese nutzen exklusiv das Force.com Web Services API, arbeiten außerhalb der Kontrolle der Force.com-Plattform und haben eine von salesforce.com getrennte Benutzeroberfläche.

Zurück zum Beispiel: Abbildung 2 zeigt die zu entwickelnde Lösung im Überblick. Zusammengefasst lässt sich diese wie folgt beschreiben: Von einem externen System (z. B. einer Integrationsplattform) werden Daten in der Force.com-Plattform abgelegt. Diese Daten, z. B. der Zustand der einzelnen Konvertierungsaufgaben, sollen integriert im normalen Benutzerinterface angezeigt werden.

Den besten Einstieg bietet die Controller-Logik. Hier werden einerseits die externen Informationen aus der Datenbank gezogen und andererseits die Benutzeroberfläche mit den aufbereiteten Daten versorgt. In Listing 1 ist ein Ausschnitt des Controllers zu sehen. Die wichtigste Aufgabe besteht in der Bereitstellung der anzuzeigenden Daten (*getStatus()*) aus einem *Force.com*-Objekt (*ConversionInstance__c*). Die Informationen, die sich im Objekt *ConversionInstance__c* befinden, wurden von einem weiteren Prozess bereitgestellt (hier nicht abgebildet).

Leicht zu erkennen ist, dass nach dem Erzeugen und Initialisieren des Controllers die Felder *finished*, *running* und *failed* initialisiert werden. Dort befindet sich die Anzahl der ausgeführten Integrationsaufgaben mit ihrem entsprechenden Status. Nachdem nun bekannt ist, wo die Daten liegen und wie sie mittels eines Controllers interessierten Parteien zur

Listing 1

Der Controller(Apex Code)

```
public class wwwArlanisComUdcController20 {

/******* VARIABLES *******/

Integer finished = 0;
Integer running = 0;
Integer failed = 0;

/******* CONSTRUCTOR ********/

public wwwArlanisComUdcController20(ApexPages.
                        StandardController controller)
{
 this.contact = (Contact)controller.getRecord();
 getStatus();
}

/****** FUNCTIONS ******/

/* handles conversion status – finished / running /
                                          failed */
public void getStatus()
{
 List<ConversionInstance__c> convInst_PieChart_
        Finished = New List<ConversionInstance__c>{};
 convInst_PieChart_Finished = [Select Id, convInstanceId
        __c, ExceptionMessage__c, EndTime__c,
StartTime__c, Status__c, convInstanceName__c,
                                          Filename__c
   From ConversionInstance__c
   Where Status__c =: 'Finished'];

 if(convInst_PieChart_Finished.size() == null)
  finished = 0;
 else
  finished = convInst_PieChart_Finished.size();
(...)
}

/****** SETTER/GETTER ******/

public Integer getFinished(){return finished;}
public void setFinished(Integer newFinished){finished =
                                          newFinished;}
(...)
}
```

Verfügung gestellt werden können, ist der nächste Schritt die Entwicklung der Benutzeroberfläche. Dazu wird das in der Plattform enthaltene komponentenbasierende Visualforce-Framework genutzt. In Listing 2 ist der entsprechende Quelltext abgebildet.

Auch hier ist der Aufbau gut erkennbar. Zuerst wird der vorher definierte Controller angegeben, um die Seite mit Daten zu versorgen. Dazu können direkt die dort vereinbarten und initialisierten Felder ausgelesen werden (*{!finished}*). Ganz wie es der Cloud-Welt entspricht, werden für die Visualisierung in der Benutzeroberfläche nicht nur die Bordmittel genutzt, sondern es wird noch eine weitere Wolke mit eingebunden – Google Visualization [4]. Mit dem Objekt *google.visualization.DataTable* kann eine ansprechende Grafik erzeugt werden. Das Ergebnis zeigt Abbildung 3 rechts oben. Auch hier sieht man, dass die Anwendung in die Menge der Anwendungen auf der Plattform integriert ist.

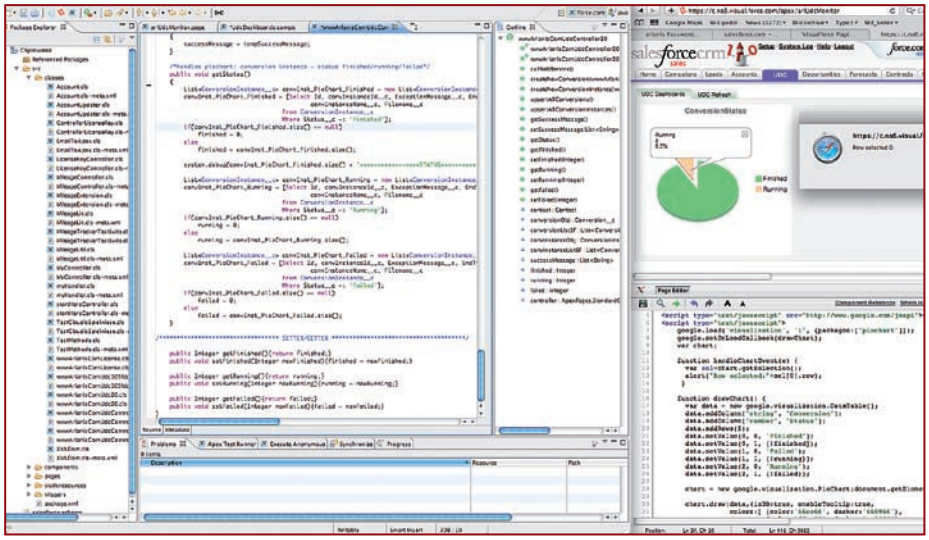

Abb. 3: Die fertige Anwendung und ihre Entwicklung in der Wolke

Fazit

Seit nunmehr zehn Jahren entwickelt sich Cloud Computing, also die Summe der „As a Service"-Konzepte sehr erfolgreich. Wie der Artikel zeigt, wurde eine Anwendung von Anfang an in der Cloud geschrieben, wo sie jetzt auch genutzt werden kann. Dass das so einfach möglich ist, hat nicht zuletzt den Grund, dass viele vorgefertigte Konzepte und auch Funktionalitäten der Force.com-Plattform nachnutzbar sind. Alle Einzelheiten zu beschreiben, würde den Rahmen eines Artikels sprengen. Interessenten sehen sich das Konzept am besten selbst an – im Fall der Force.com-Plattform kann unter http://www.force.com ein freier Testaccount eingerichtet und genutzt werden.

Listing 2

Die Benutzeroberfläche (Visualforce)

```
<apex:page id="UdcMonitor"
 standardController="Contact" extensions=
 "wwwArlanisComUdcController20" sidebar="false"
 standardStylesheets="false">
<HTML>
 <HEAD>
<script type="text/javascript" src="http://www.google.
                                    com/jsapi"></script>
 <script type="text/javascript">
  google.load('visualization', '1',
                                {packages:['piechart']});
  google.setOnLoadCallback(drawChart);
  var chart;

  function drawChart() {
   var data = new google.visualization.DataTable();
   data.addColumn('string', 'Conversion');
   data.addColumn('number', 'Status');
   data.addRows(5);
   data.setValue(0, 0, 'Finished');
   data.setValue(0, 1, {!finished});
  (...)
   chart = new google.visualization.PieChart(
            document.getElementById('visualization'));
   chart.draw(data,{is3D:true, enableTooltip:true,
            colors:[ {color:'66cc66', darker:'669966'},
      {color:'ffcc99', darker:'cc9966'}, {color:'#ff6666',
                                    darker:'cc6666'}],
 legend:'right',
     title:'ConversionStatus',
     titleColor:'666666'});

   google.visualization.events.addListener(chart,
                          "select", handleChartEvent);
  }
 </script>
(...)

</HEAD>
<BODY>
 <apex:panelGroup >
  <div id="visualization" style="width: 300px;
                                height: 300px;"></div>
 </apex:panelGroup>
</BODY>
</HTML>
</apex:page>
```

Andreas Holubek *arbeitet als VP Engineering für die arlanis Software AG und blickt auf eine langjährige Arbeit in Konzeption, Design und Programmierung von SW-Lösungen zurück. Bei arlanis ist er für die UDC-Produktfamilie sowie die SOA und EAI-Strategien verantwortlich. Sein besonderes Interesse gilt der Datenintegration, serviceorientierten Architekturen und den verschiedenen Web-Service-Techniken. Er ist bekannt als Autor und Referent auf verschiedenen Konferenzen. Daneben beschäftigt er sich mit der Einführung und Umsetzung von modernen Technologien und Spezifikationen in der Praxis.*

Claudia Matthias *arbeitet in den Bereichen Design und Entwicklung bei der arlanis Software AG. Den Schwerpunkt ihrer Arbeit bilden Aufgaben im Bereich Salesforce.com und des User Interface. Weiterhin beschäftigt sie sich mit der Umsetzung von Design- und Webentwicklungsaufgaben.*

Links & Literatur

[1] Salesforce.com: http://www.salesforce.com

[2] Salesforce.com CRM: http://www.salesforce.com/de/products

[3] Force.com-Entwicklerseite: http://developer.force.com/

[4] Google Visualization Package: http://code.google.com/intl/de-DE/apis/visualization