entwickler.press
shortcuts

Apps mit Azure

Roman Schacherl, Marc André Zhou,
Felix Mokross, Thomas Sobizack

Roman Schacherl, Marc André Zhou, Felix Mokross, Thomas Sobizack

Apps mit Azure

ISBN: 978-3-86802-565-1

Ein Imprint der Software & Support Media GmbH

1 Web, Mobile, API und Logic in der Cloud

Wer bisher einen einfachen Einstieg in die Welt von Microsoft Azure suchte, war mit Azure-Websites gut bedient: das Hosten von Webanwendungen, angereichert mit Cloud-Features wie Scale-up oder Scale-out. Im März 2015 wurden Websites als Teil der neu vorgestellten App-Services umbenannt in Web-Apps – und erweitert um teils bekannte (Mobile-Apps) und teils komplett neue Komponenten (Logic-Apps und API-Apps). Hier ein erster Überblick.

Microsoft treibt die Entwicklung von Azure mit hoher Geschwindigkeit voran. So gut wie jeden Monat werden neue Ankündigungen veröffentlicht. Selbst intensive Nutzer sind herausgefordert, wenn sie an allen Fronten Bescheid wissen wollen – von Azure-Anfängern gar nicht zu sprechen. Nicht gerade dienlich war dabei die bisherige Vorgehensweise, alle Features quasi gleichwertig nebeneinander zu positionieren. Da gab es Websites, virtuelle Maschinen und SQL-Datenbanken gleichgestellt mit HDInsight, Media-Services und Machine Learning. Welche Komponenten gemeinsam Sinn ergeben und zusammenspielen, war schwer ersichtlich; schon gar nicht aber konnten verschiedene Azure-Services innerhalb derselben Instanz betrieben werden. Wer beispielsweise einen Mobile-Service und eine Website betrieben hat, musste getrennt dafür zahlen.

Mit dem neu vorgestellten App Service wird versucht, vier zusammenhängende Dienste unter einem gemeinsamen Namen anzubieten. Eine Grundbedingung aus der Marketingabteilung war offensichtlich die Verwendung des Wortes App im Namen: So wurden die Websites zu Web-Apps, die Mobile-Services zu Mobile-Apps und zwei Neuzugänge auf Logic-App bzw. API-App getauft. Jeder dieser Services kann weiterhin separat und

unabhängig genutzt werden – doch es gibt eine Story, wie die genannten Komponenten zusammenspielen können und damit vielleicht ein leichterer Einstieg in die Welt von Microsoft Azure gegeben werden kann.

Web-Apps

Aber der Reihe nach. Am wenigsten spektakulär verlief die Ankündigung der App-Services für die Nutzer von Azure-Websites. Die Umbenennung blieb die einzige direkt sichtbare Auswirkung, Web-Apps können nicht mehr und nicht weniger als die bisherigen Websites. Sämtliche bereits deployten Anwendungen waren vom ersten Tag an unter dem neuen Begriff sichtbar, kein Migrationsaufwand, keine Probleme. Der Mehrwert kommt erst durch die optionale Nutzung der weiteren Services, die in einem bestehenden Webcontainer ohne zusätzliche Kosten betrieben werden können.

Mobile-Apps

Unter dem Namen Mobile Services waren schon bisher mehrere Angebote vereint, die das Entwickeln von Apps erleichtern konnten: Ein REST-basierter Zugriff auf Daten, das Senden von Push Notifications quer durch alle Plattformen oder die Authentifizierung mittels Providern wie Facebook, Twitter oder ein Microsoft-Account. Zum Teil ist es auch möglich, die Services durch eigenen Backend-Code anzupassen: Die REST-Datenschnittstelle stellt zum Beispiel Erweiterungspunkte zur Verfügung, die ähnlich wie Datenbank-Trigger genutzt werden können. Als Programmiersprachen für das Backend stehen Node. js oder .NET zur Verfügung. Die gesamte Funktionalität ist über eine REST-basierte Schnittstelle zu konsumieren, es gibt aber SDKs, um den Zugriff aus Windows Phone, iOS und Android zu erleichtern.

Im neuen App-Service-Paket tauchen nun große Teile der Funktionalität unter dem Namen Mobile-Apps wieder auf – aber nicht ganz so reibungslos und unverändert wie bei den

zuvor genannten Websites. Erstes Anzeichen: Bisherige Mobile Services wurden nicht automatisch migriert, im alten Azure-Portal befinden sich weiterhin alle Instanzen unter unverändertem Namen (eine Anleitung zur Migration befindet sich unter [1]). Mit ein Grund kann sein, dass als Sprache für das Backend in Mobile-Apps vorerst nur .NET zur Verfügung steht – die in den Mobile Service gebräuchlichere Node.js-Variante fehlt noch. Der für das Backend notwendige Code kann nun aber gemeinsam mit der Web-App ohne zusätzliche Kosten im selben Container gehostet werden – inklusive Auto-Scaling, Traffic-Manager-Support oder Deployment-Vorteilen wie Continuous Integration und Staging Deployments. Als weitere Neuerung sind Verbindungen zu On-Premise-Datenbanken möglich, ein eingerichtetes virtuelles Netzwerk mit Point-to-Site-VPN wird vorausgesetzt. Wer vor der Aufgabe steht, eine mobile App für iOS, Android oder Windows Phone zu entwickeln und dafür Push Notifications, Authentifizierung oder (On-Premise)-Datenzugriff benötigt, sollte sich Mobile-Apps ansehen.

Es ist anzunehmen, dass die bisherigen Mobile Services über kurz oder lang komplett durch die neuen Mobile-Apps ersetzt werden, für den Moment gibt es allerdings beide Varianten parallel.

API-Apps

Kommen wir zu den Neuerungen: API-Apps sind brandneu und fügen der Azure-Palette wieder einen deutlichen Mehrwert hinzu. Im Wesentlichen handelt es sich dabei um Web-API-Projekte, die um Metadaten und Authentifizierung angereichert sind.

Als Format für diese Metadaten kommt – ganz im Sinne des aktuellen Microsoft-Mindsets – keine hauseigene, proprietäre Lösung, sondern ein bewährtes, existierendes Open-Source-Projekt namens Swagger [2] zum Einsatz. Eine Swagger-Definition besteht aus einer Beschreibung der einzelnen

Methoden und deren Parameter in einem spezifizierten JSON-Format. Alter Wein in neuen Schläuchen? CORBA? WSDL? Im Prinzip ja. Das Swagger-Format ist aber sehr schlank und erfordert keine Änderung von bestehenden APIs – theoretisch könnte die Swagger-Definition sogar von jemand anderem als dem API-Hersteller geschrieben werden. Die größten Vorteile, die Metadaten mit sich bringen, sind schnell beschrieben: automatische Dokumentation und Client-SDK-Generierung.

Hello, API-App

Wie sieht das in der Praxis aus? App-Services sind bereits seit dem Azure-SDK 2.5.1 integriert. Beim Erstellen einer neuen ASP.NET-Webapplikation steht als Template „Azure-API-App-(Preview)“ zur Verfügung, dadurch wird ein Web-API-Projekt mit einigen zusätzlichen Referenzen erstellt (**Abb. 1.1**). Sollten Sie ein bestehendes ASP.NET-Web-API-Projekt verwenden wollen, klicken Sie mit der rechten Maustaste auf das Projekt und anschließend auf ADD | Azure API App SDK. Sie werden anschließend gefragt, ob die Swagger-Metadaten automatisch erstellt werden sollen (**Abb. 1.2**, die Antwort lautet: Ja).

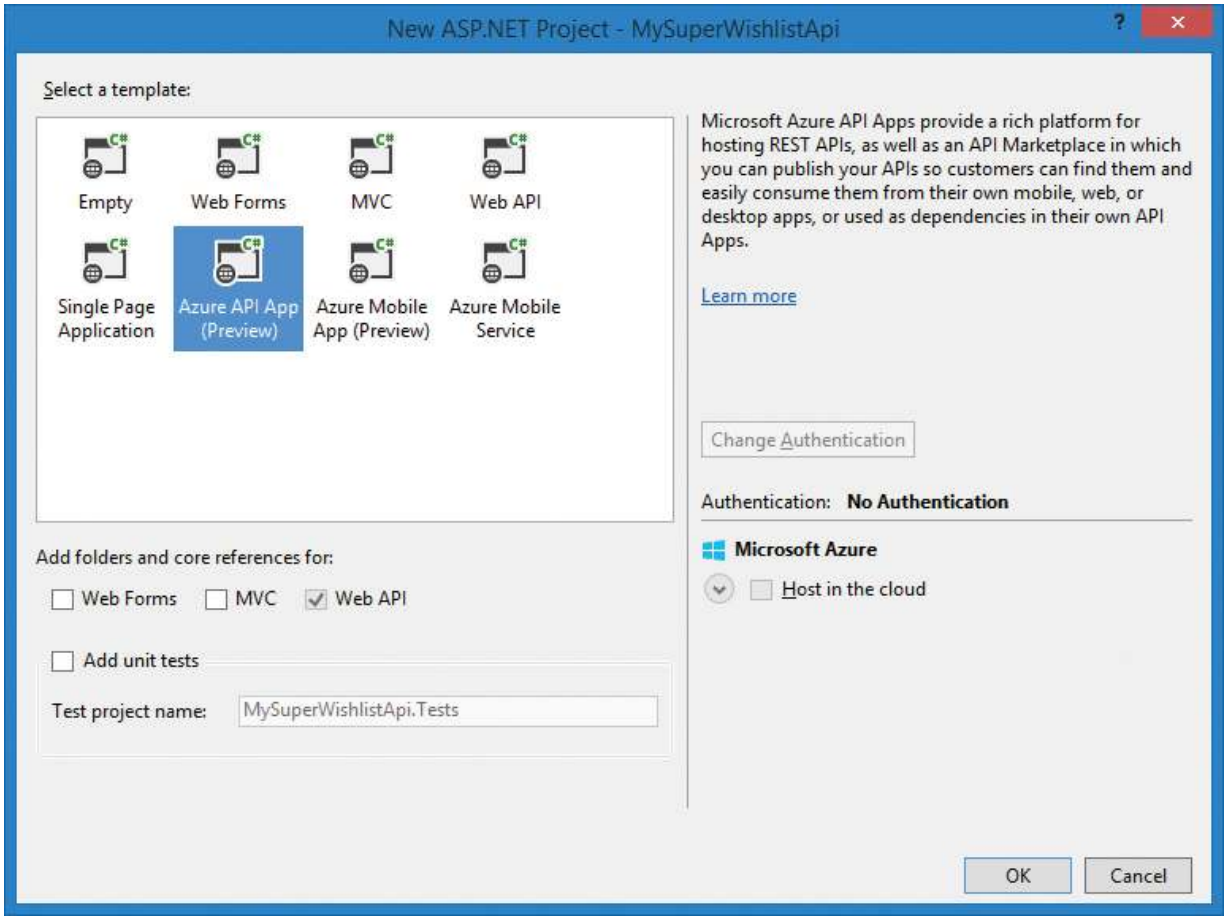

Abbildung 1.1: Erstellen eines Azure-API-App-Projekts in Visual Studio

Abbildung 1.2: Metadatengenerierung oder -auswahl

Das Erstellen der Web-API-Controller unterscheidet sich nicht von der gewohnten Art und Weise; wie bisher leiten Sie von *ApiController* ab, erstellen darin Actions und verwenden ggf. Routing-Attribute (Listing 1.1).

```
public class WishlistController : ApiController
{
  public string Get()
  {
    return "Hello API App!";
  }

  public async Task Post(WishRequest
request)
  {
    // store data
  }
}
```

Listing 1.1: Web-API-Controller

Was Ihr Web-API nun zu einer API-App macht, sind die Metadaten. Die automatische Metadatengenerierung sollte aktiviert sein; erstes Erkennungsmerkmal ist die Datei *apiapp.json* im Hauptverzeichnis mit einigen allgemeinen Metadaten. Wichtiger ist die Klasse *SwaggerConfig* im Unterordner *App_Start*. Hier können Sie in die Generierung der Metadaten eingreifen und beispielsweise festlegen, ob auch das Swagger-UI bereitgestellt werden soll: ein interaktives UI zum Testen des API.

Wenn Sie die Anwendung ausführen, können Sie den Erfolg unmittelbar testen: Unter *http://localhost:<port>/swagger/docs/v1* können Sie das Swagger-JSON-Format ansehen, unter */swagger* erscheint das UI (**Abb. 1.3**).