ALEXA SKILL MIT EINEM PHP BACKEND REALISIEREN

Skill-Backend

Mit einfachen Bordmitteln die ersten Schritte in die Welt von Alexa unternehmen.

Amazon treibt den Ausbau seines Alexa-Ökosystems sehr aggressiv voran. Von der Öffentlichkeit am ehesten wahrgenommen wird die massive Expansion der Alexa unterstützten Endgeräte. Auch jeder Nutzer erfährt wöchentlich per Mail, was Alexa alles Neues kann. Dabei erwähnt Amazon auch oft neue Rekordmarken über die Anzahl verfügbarer Skills - weil wir Drittentwickler fleißig waren.

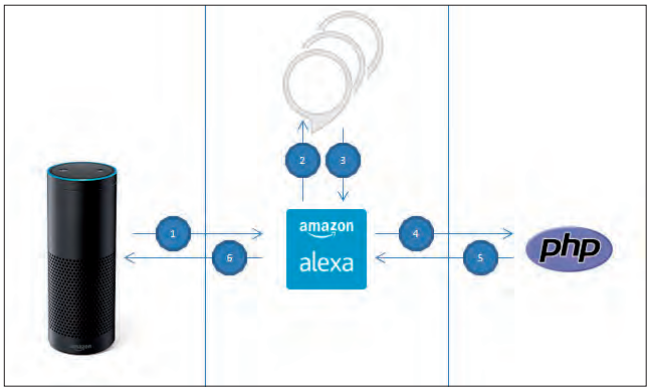
Diese Front überlässt Amazon nicht dem Zufall. Der Kampf der Smartphone-Ökosysteme hat gezeigt, wie ausschlaggebend ein breites Angebot an Drittanwendungen ist. Neben dem Pflichtprogramm einer guten Online Dokumentation und der Bereitstellung zahlreicher Tutorials, glänzt Amazon aber auch mit einem intensiven Community-Building. Seit Monaten tourt ein spezielles Team (auch) durch deutsche Städte und führt in einer Tagesveranstaltung die Teilnehmer durch die ersten Tutorials und hilft live beim Überwinden der ersten Hürden. Zur Motivation das erworbene Wissen auch produktiv einzusetzen wurden großzügig Goodies wie T-Shirts, Hoodies, Socken und sogar Endgeräte ausgelobt, wobei im Laufe des letzten Jahres die Bedingungen für den Erhalt der Hardware verschärft wurden.

Anatomie eines Skills

Das Umsorgen von uns Entwicklern zahlt sich für Amazon aber auch noch in anderer Hinsicht aus. Der Schnelleinstieg erfolgt nämlich unter Zuhilfenahme der Amazon Web Services (AWS). Zwar hat Amazon auch ein Sponsoring-Programm aufgelegt, um die Kosten für das Hosting eines Skills auf der Amazon-Cloud-Infrastruktur zu kompensieren - die Hoffnung ist aber, wer einmal angefixt ist wechselt nicht mehr so leicht zu anderen Anbietern.

Für bereits etablierte Projekte mit eigener Software und Server-Infrastruktur stellt sich bei der Realisierung eines Alexa-Skills die Frage genau umgekehrt. Wie viel der bestehenden Infrastruktur und des bestehenden Codes kann wiederverwendet werden und was wird zusätzlich gebraucht? Aus diesem Grund wird hier nicht einfach ein Tutorial oder fertiges Framework verwendet, sondern wir arbeiten uns Schritt für Schritt durch die minimale Implementierung eines Skills.

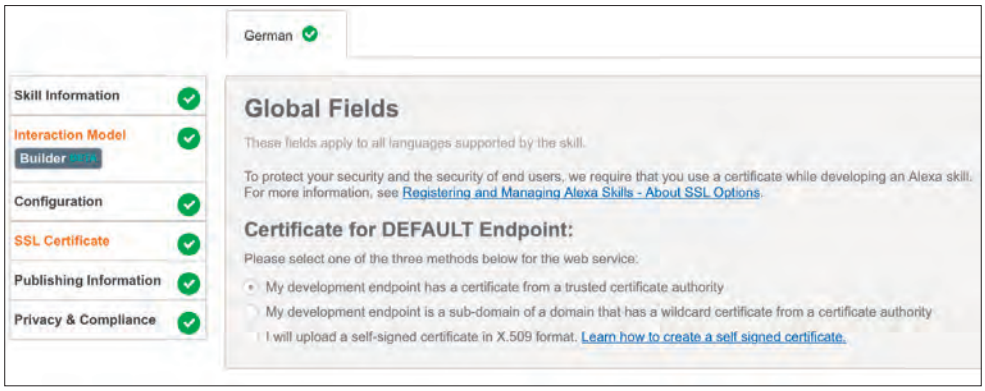
Hierzu ist zunächst einmal das Verständnis über die grobe Funktionsweise eines Skills notwendig (Bild 1). Die Alexa Geräte an sich stehen dabei außerhalb des Einflussbereiches eines Entwicklers. Hier läuft ein Betriebssystem, welches ständig lauscht und auf das sogenannte Wake Word (in der Regel »Alexa«) wartet. Sobald dieses erkannt ist, wird die Anfrage aufgenommen und an die Alexa Backendsysteme geschickt. Diese wandeln die Sound Daten in Text um (VoiceToText - VTT). Anschließend wird im Text nach sogenannten Invocation Names eines vom Nutzer aktivierten Skills gesucht. Dabei gilt, dass der Invocation Name aus mindestens zwei Wörtern bestehen soll. Eine Ausnahme bilden spezielle Wörter – in der Regel Produktnamen - für deren Verwendung man jedoch im Besitz der Markenrechte sein muss. Im nächsten Schritt wird anhand der bei Amazon hinterlegten Definition des Skills untersucht, welcher Intent (vergleichbar mit einem



Ablaufdiagramm eins Skill-Aufrufes (Bild 1)

Methodenname) vom Nutzer gemeint ist. Dies erkennt das Alexa Backend anhand von durch den Entwickler zu hinterlegenden Beispielsätzen (Sample Utterances), die recht dumpf durchsucht werden - kleinere Abweichungen vom Nutzer können hier schon verhindern, dass eine richtige Zuordnung erfolgt.

Erst jetzt wird eine JSON-Datei erzeugt, die an das hinterlegte Skill-Backend geschickt wird. Dieses verarbeitet die Anfrage und antwortet ebenfalls mit einem JSON aus welchem dann die Soundausgabe generiert wird (TextTo-Voice - TTV). Es gibt auch Möglichkeiten, die generierte Ausgabe mittels Speech Synthesis Markup Language (SSML) genauer vorzugeben beziehungsweise sogar vorgefertigte Sounddateien direkt auszugeben, was in diesem Artikel jedoch nicht weiter betrachtet wird.

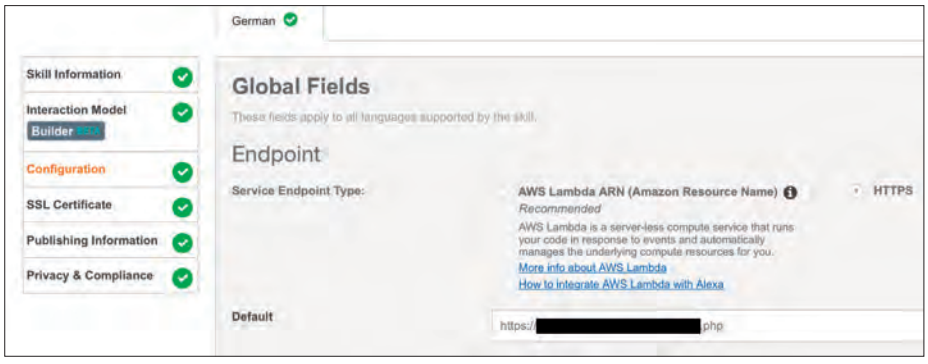


SSL Certificate-Bereich der Alexa Skill Konfiguration (Bild 3)

Eigener PHP-Server als Skill-Backend

Um ein PHP Script als Backend für die Verarbeitung der JSON-Anfrage und die Generierung der entsprechenden Antwort zu nutzen zu können, muss dieses über eine eigene URL erreichbar sein. Diese URL muss mit einem trusted SSL Zertifikat abgesichert sein, sodass diese per HTTPS erreichbar ist. In vielen kleinen Hosting-Angeboten ist mittlerweile meist ein entsprechendes SSL Zertifikat kostenfrei enthalten. Alternativ bietet Amazon auch die Möglichkeit ein privat erstelltes Zertifikat zu nutzten. Allerdings ist dies nur für den eigenen Test ausreichend.

Davon ausgehend, dass ein Amazon-Entwickleraccount vorhanden ist, muss nun in *Developer Console* unter dem Reiter *Alexa* die Option *Alexa Skill Kit* ausgewählt werden. Hier wird die Definition eines Skills angelegt, wobei die ersten beiden Bereiche *Skill Information* und *Interactive Modells* unabhängig von der Art des Backends vorgenommen werden und deshalb in diesem Artikel übersprungen werden. Unter *Configuration* weichen wir nun von den Standard-Tutorials ab.



Configuration-Bereich der Alexa Skill Konfiguration (Bild 2)

Zuerst ändern wir den *Service Endpoint Type* von AWS auf HTTPS. Danach können wir eine Default-URL des PHP-Skripts angeben (Bild 2). Alternativ können wir je geographischer Region der Alexa-Nutzer eine eigenständige URL angeben. Dies ist zum Beispiel aus Gründen der Latenzreduzierung sinnvoll, sofern die URLs entsprechend der Unterteilung von Amazon (Nord-Amerika, Europa, Indien, Asien) geographisch gehostet werden können.

Im vierten Bereich *SSL Certificate* muss noch einmal eingestellt werden, welche Art von Zertifikat genutzt wird. Zwar ist hier auch das Hinterlegen eines privat erzeugten SSL Zertifikates möglich (Bild 3), Amazon legt in seiner Dokumentation »Security Testing for an Alexa Skill« unter 2.2 jedoch fest, dass für einen eigenen Endpunkt bei der Freigabe ein Trusted Authority Zertifikat (zum Beispiel vom Hosting Anbieter bereitgestellt) abgeprüft wird. Zur Vollständigkeit sei noch erwähnt, dass der definierte Port 443 genutzt werden muss.

Alexa fragt an

Nachdem die Grundvoraussetzungen nun erfüllt sind, kann mit der Entwicklung der Skill Logik begonnen werden. Im folgenden Listing ist einmal das minimale JSON angegeben, durch das Alexa mit »Hallo Welt« antwortet:

```
{
  "version": "1.0",
  "response": {
      "outputSpeech": {
                  "type": "PlainText",
                  "text": "Hallo Welt."
          },
          "shouldEndSession": true
    },
    "sessionAttributes": {}
}
```

Die zu generierende Antwort lässt sich durch zahlreiche Parameter konfigurieren, allerdings wollen wir uns erst einmal darauf konzentrieren, wie wir die Anfrage auswerten können, um danach die ▶

Listing 1: Alexa Anfrage prüfen

```
$requestOK=0;
$http_header = getallheaders();
$amazon_url = $http_header["Signaturecertchainurl"];

$urlArray = parse_url($amazon_url);

if (strtolower($urlArray["scheme"]) != "https")
    $requestOK=1;
if (strtolower($urlArray["host"]) !=
"s3.amazonaws.com")
    $requestOK=2;
if (isset($urlArray[„port"]) &&
($urlArray[„port“] != 443))
    $requestOK=3;

//Normalisieren von Realtiv-Pfaden
$tempA = explode("/", $urlArray["path"]);
array_splice($tempA, 0, 1);
for ($i = 0; $i < count($tempA); $i++)
  {
    if ($tempA[$i] == "..")
      {
        array_splice($tempA, $i, 1);
        if ($i>0)
          {
            $i--;
            array_splice($tempA, $i, 1);
          }
        $i--;
      }
    }

if (strtolower($tempA[0]) != "echo.api")
    $requestOK=4;
```

richtige Antwort zu liefern. Hierzu müssen wir die Anfrage von Alexa zunächst in ein Objekt laden:

```
// Den Datenstream der Anfrage erhalten
$postPlain = file_get_contents( ‚php://input‘ );

// Decode the JSON into a stdClass object
$alexa_request = json_decode( $postPlain );
```

Anschließend kann man einfach mittels

```
$alexa_request->request->intent->name
```

auf einen String zugreifen, der den Namen des aufzuführenden Intents enthält (case sensitiv). Mit einer *switch*-Anweisung kann man nun einen einfachen Dispatcher schreiben, um die richtige Logik zum Generieren der gewünschten Antwort aufzurufen.

Default-Intents von Amazon

An dieser Stelle sei erwähnt, dass es Default-Intents von Amazon gibt, welche auf jeden Fall unterstützt werden müssen, um später eine Freigabe zu erhalten. Diese sind *AMAZON.CancelIntent*, *AMAZON.HelpIntent* und *AMAZON.StopIntent*. Zwar ist man frei in der Vergabe der Intent-Namen, aber an den Namen der Default-Intents kann man bereits erkennen, dass *AMAZON.* als Zeichenfolge reserviert ist. Neben der reinen Auswahl der Funktion, kann Alexa aber auch Parameter vom User übermitteln. Diese müssen in der Intent-Definition als Platzhaltern (sogenannte Slots) konfiguriert sein (Bild 4). Ein Zugriff auf die Parameter ist mit

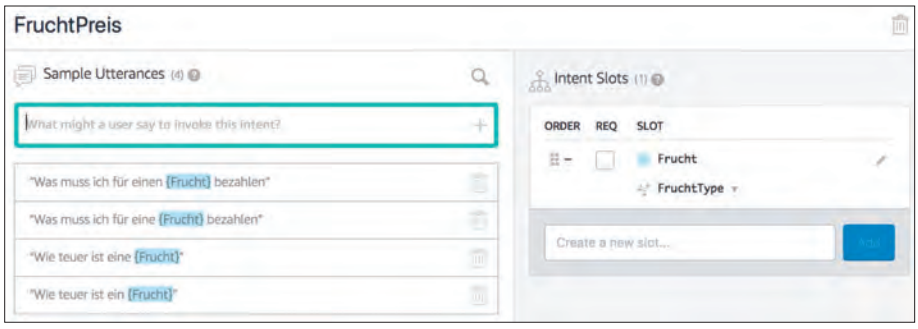
```
$alexa_request->request->intent->slots->Frucht->value
```

möglich, wobei *Frucht* durch den jeweiligen Slot-Namen ersetzt werden muss.

Wichtig zu wissen ist, dass der Wert dem entspricht, was an der Stelle des Platzhalters im Beispielsatz vermeintlich erkannt wurde. Datentypen - auch eigens durch String-Auflistungen definierte - stellen dabei nur Hilfen für die VTT-Funktion dar, um das Gesagte besser erkennen zu können.

Trotzdem ist es möglich, dass ein Slot der über einen eigenen Datentyp (zum Beispiel *FruchtType*) durch die Auflistung der Werte *Apfel, Birne* und *Tomate* definiert wurde, auch den Wert *Gurke* oder *Grossmutter* liefert, wenn die VTT Funktion dies in dem Satzstück erkannt hat (Bild 5). Über diesen Trick ist es bisher möglich, dass Skills Freitext zum Beispiel zum Twittern und Chatten erfassen können.

In der Definition eines Datentyps lassen sich für einen Wert auch Synonyme auflisten. Auch hier ist es wichtig, dass beim Erkennen eines synonymen Begriffs, dieser (zum Beispiel *Malus*) anstelle des Haupt-



Intent Definition mit Slot (Bild 4)

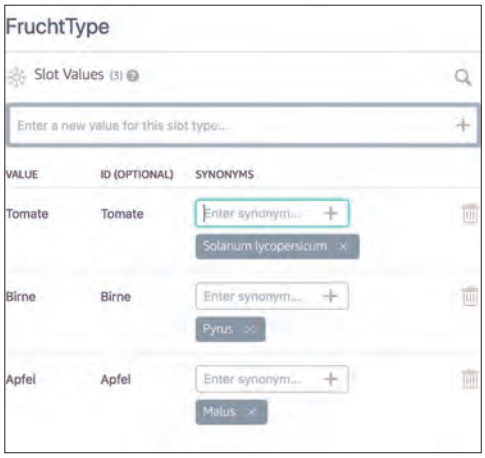
begriffes (zum Beispiel *Apfel*) übergeben wird. Wenn der konkrete Begriff nicht von Bedeutung ist, kann man auch mit der optionalen ID arbeiten.

Eine ID kann unabhängig vom Hauptbegriff definiert werden, oder wie im Beispiel auch genauso. Die ID eines erkannten Begriffes beziehungsweise seiner Synonyme lässt sich mittels

```
$alexa_request->request->intent
->slots->Frucht->resolutions
->resolutionsPerAuthority[0]
->values[0]->value->id
```

abfragen. Zum Experimentieren sollten diese Grundlagen erst einmal ausreichen.

Es bietet sich an, sein Skill möglichst bald auf einem echten Endgerät zu testen, da der Web-Simulator (noch) nicht alle Feinheiten (zum Beispiel die gerade erwähnte Resultions) unterstützt. Dies ist möglich, wenn der Account des Nutzers, mit welchem das Endgerät eingerichtet ist, dem Account der Entwicklers entspricht oder von diesem über den *Beta Test*-Button in der Skill-Konfiguration eingeladen wurde.



Slot Type Definition mit IDs und Synonymen (Bild 5)

Zum Veröffentlichen des Skills wird es allerdings noch nicht ausreichen. Amazon hat eine längere Liste an Anforderungen, wie zum Beispiel Sicherheitsprüfungen, welche zwar keine direkte Auswirkungen auf die Skill-Logik beziehungsweise deren Test haben, jedoch ohne deren Erfüllung Amazon das Skill nicht in die freie Wildbahn entlässt.

Das obligatorische SSL-Zertifikat und die verpflichtende Nutzung des Ports 443 wurden bereits erwähnt. Darüber hinaus muss das Skill auch validieren, dass die Anfrage tatsächlich von Amazon kommt. Hierzu gehört auch das Ablehnen von nicht signierten Anfragen.

Dafür holen wir zunächst aus dem http-Header die URL des Amazon-Zertifikates und überprüfen die URL entsprechend den Vorgaben (Listing 1). Während der Zertifizierung des Skills überprüft Amazon auch diverse ungültige URLs, die man ablehnen muss, um die Zertifizierung erfolgreich zu bestehen:

```
// $requestOK siehe Listing 3
if ($requestOK!=0)
```

▶

```
  {
    http_response_code(400);
    $protocol = (isset($_SERVER['SERVER_PROTOCOL']) ?
    $_SERVER['SERVER_PROTOCOL'] : 'HTTP/1.0');
    header($protocol . ' 400 Bad Request');
    die();
}
```

Anschließend laden wir das Zertifikat und überprüfen, ob es (zeitlich) gültig ist:

```
$certOK=0;
// $amazon_url siehe Listing 3
$cert = file_get_contents($amazon_url);
openssl_x509_parse($cert);

if( $certinfo['validFrom_time_t'] > time() ||
$certinfo['validTo_time_t'] < time() )
    $certOK=1;
```

Die Implementierung dieser Prüfung wird von Amazon vorgeschrieben, jedoch momentan im Rahmen der Zertifizierung nicht nachgeprüft.

Abschließend validieren wir nun, ob die Signatur im HTTP-Header wirklich der Signatur der Anfrage zu diesem Zertifikat gehört:

```
$signature = base64_decode($http_header["Signature"]);

// $cert siehe Listing 5
$request = file_get_contents( 'php://input' );
if (openssl_verify($data, $signature, $cert) != 1)
    $certOK=2;
```

Verläuft dieser Test negativ, verbieten wir den Zugriff:

```
// $certOK siehe Listing 5

if ($certOK!=0)
  {
    http_response_code(403);
    $protocol = (isset($_SERVER['SERVER_PROTOCOL']) ?
    $_SERVER['SERVER_PROTOCOL'] : 'HTTP/1.0');
    header($protocol . ' 403 Forbidden');
    die();
}
```

Damit ist sichergestellt, dass die Anfrage von Amazon kommt. Sie sollten allerdings auch sicherstellen, dass die Anfrage tatsächlich von Ihrem Skill kommt, ansonsten könnte jemand anderes, der herausgefunden hat, wie Ihr Endpunkt heißt, einfach in seinem Namen ein Skill veröffentlichen und bei Ihnen als Trittbrettfahrer Last erzeugen. Hierzu kann die Skill-ID, welche Amazon beim Anlegen der Skill-Konfiguration vergibt mit

```
if ( ‚amzn1.ask.skill.XXXXXXXXXX‘ !=
$alexa_request->session->application->applicationId)
{
 $requestOK = 5;
}
```

abgefragt werden. Die letzte von Amazon vorgeschriebene Sicherheitsprüfung wird zurzeit nicht während der Freigabe geprüft, sollte aber schon jetzt berücksichtigt werden: eine Anfrage darf nicht älter als 150 Sekunden sein. Hierzu enthält die Anfrage einen ISO 8601 Zeitstempel, der mit

```
if ( strtotime($alexa_request->request-> timestamp) > (
time() + 150) ) { $requestOK = 6; }
```

überprüft wird.

Fazit

Mit wenigen Zeilen Code kann man die PHP-Logik bestehender Projekte für ein Amazon Alexa Skill wieder verwenden. Von hier aus lässt sich die Antwort entsprechend der Amazon-Dokumentation für die verschiedenen Endgeräte-Typen (zum Beispiel Echo Show mit Bildschirm) weiter ausbauen. Alternativ sollte man abwägen, ob man nicht zu einem der momentan entstehenden PHP-Frameworks greift. Hier gilt es zu prüfen, ob diese die nötige Flexibilität (zum Beispiel das Verändern der Antwort je Endgerät-Typ) erlauben und zu beobachten, ob Ihre Weiterentwicklung mit den häufigen Neuerungen durch Amazon schritt hält. ■

Links zum Thema

- Amazon Developer Portal: Request and Response JSON Reference
  https://developer.amazon.com/de/docs/custom-skills/request-and-response-json-reference.html
- PHP Library der Travello GmbH
  https://github.com/travello-gmbh
- PHP Library für Twig
  https://github.com/internetofvoice/vsms-skeleton
- PHP Library für Laravel and Lumen
  https://github.com/develpr/alexa-app
- PHP Library
  https://github.com/MiniCodeMonkey/amazon-alexa-php
- Alexa Custom Skill für das Heimautomatisierungs-Framework Framework Patami
  https://docs.braintower.de/display/IPSPATAMI

Emil Thies
ist in der IT Abteilung eines DAX-Unternehmens angestellt und probiert nebenberuflich neue Technologien aus, indem er eigene Apps, Skills und Actions programmiert und veröffentlicht.