# Alexa 1: Hello World

Welcome to the first module of Amazon Alexa and Makers Academy's short course on building Alexa skills using Ruby! Amazon's Alexa Skills Kit allows developers to extend existing applications with deep voice integration, and to construct entirely new applications that leverage the cutting-edge in voice-controlled technology.

During this course, we'll cover all the terminology and techniques required to get fully-functional skills pushed live to owners of Alexa-enabled devices all round the world. We'll be using Ruby.

## What's in this module?

This module contains a basic introduction to scaffolding a skill, and interacting with Alexa. This module introduces:

* Intent Schemas
* Utterances
* Alexa communication paradigm
* Tunnelling a local application using ngrok over HTTPS
* Connecting Alexa to a local development environment
* Alexa-style JSON requests and responses

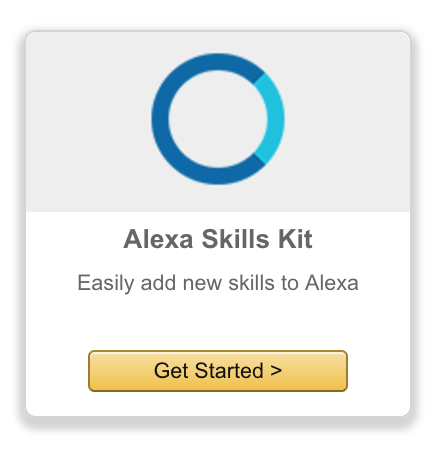
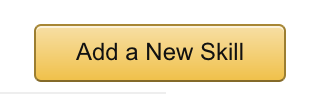
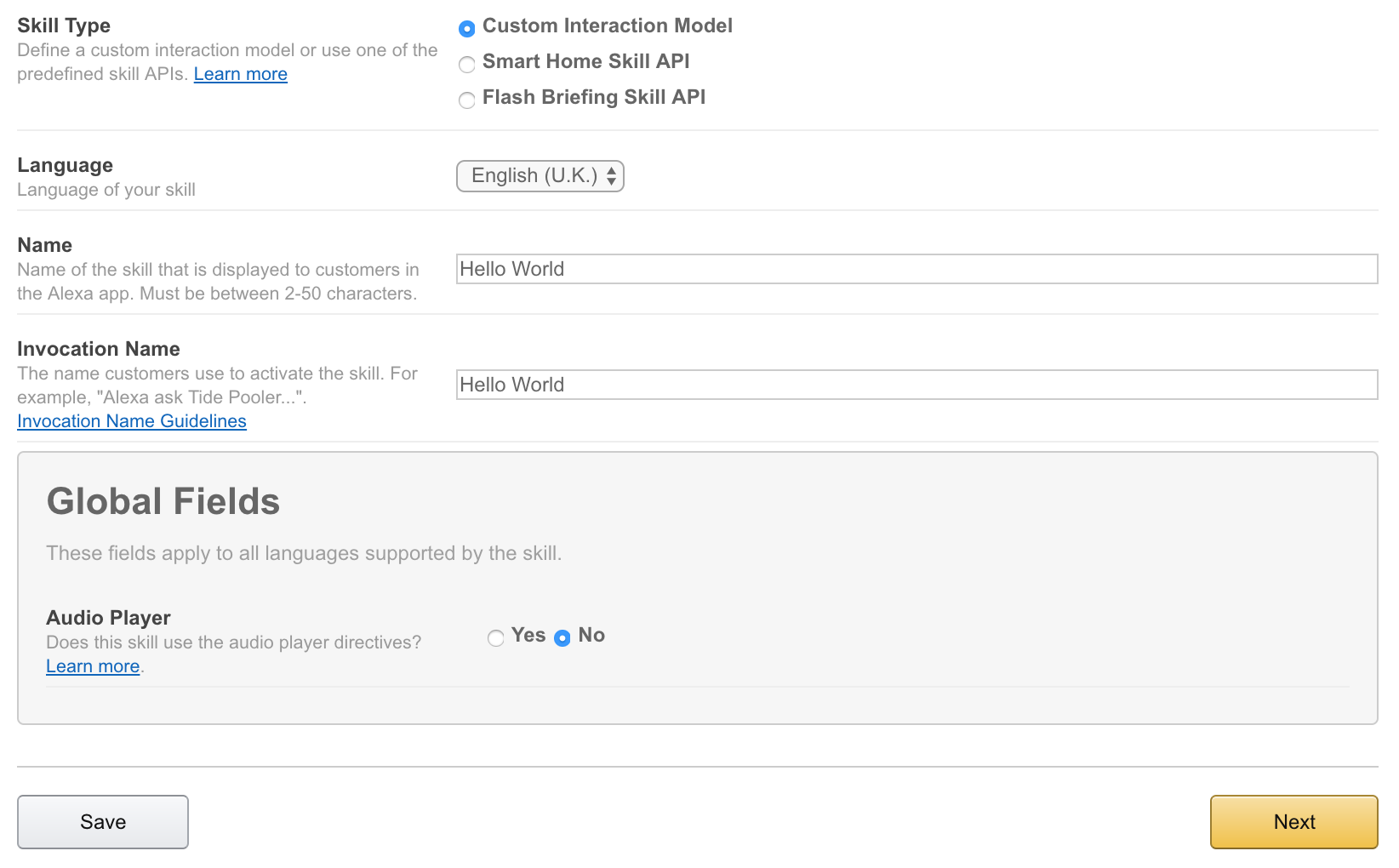
During this module, readers will construct a simple skill, called 'Hello World'. While building this skill, readers come to understand how the above concepts work and play together. This module uses:

* Sinatra
* Ruby's JSON library

Let's get started!

## 1. Amazon-side setup: setting up the Voice User Interface ('VUI')

Our first step is to set up the skill on Amazon.

* Sign up, then sign in to the [Amazon Alexa Developer Dashboard](https://developer.amazon.com/alexa)
* Click 'Alexa' on the navigation bar, then 'Get Started with the Alexa Skills Kit': 
* ‘Add a New Skill’: 
* Use a 'default custom interaction model'.
* Set up the app:
* Language
* Name (‘Hello World’)
* Invocation Name (‘Hello World’) 

The Invocation Name is used by the user to access a certain skill. For example, "Alexa, ask to say Hello World."

##### Intent Schemas

Now we have a new skill, let's construct the **Intent Schema**. > The Intent Schema lists all the possible requests Amazon can make to your application.

{  
 "intents": [  
 {  
 "intent": "HelloWorld"  
 }  
 ]  
}

The minimal Intent Schema is a JSON object, with a single property: intents. This property lists all the actions an Alexa skill can take. Each action is a JSON object, with a single property: intent. The intent property gives the name of the intent.

##### Utterances

Now we have the Intent Schema, let's make the **Utterances**. Utterances map Intents to phrases spoken by the user. They are written in the following form:

IntentName utterance

In our case, we have only one Intent: HelloWorld, and we'd like the user to say the following:

Alexa, ask Hello World to say hello world.

Our Utterances are:

HelloWorld say hello world

We've now set up our skill on Amazon's Alexa Developer Portal.

## 2. Setting up the Backend: a local Tunnelled Development Environment

Our second step is to set up our local Ruby application, ready to receive encrypted requests from Amazon’s servers (i.e. HTTP requests over SSL, or ‘HTTPS’ requests).

We will walk through setting up a Ruby server using Sinatra, running locally, and capable of receiving HTTPS requests through a Tunnel.

Alternatively, you could set up a **remote** development server using [Heroku](http://heroku.com) (with [Heroku SSL](https://devcenter.heroku.com/articles/ssl)), [Amazon Elastic Beanstalk](https://aws.amazon.com/elasticbeanstalk/?sc_channel=PS&sc_campaign=acquisition_UK&sc_publisher=google&sc_medium=beanstalk_b&sc_content=elastic_beanstalk_e&sc_detail=elastic%20beanstalk&sc_category=beanstalk&sc_segment=159760119038&sc_matchtype=e&sc_country=UK&s_kwcid=AL!4422!3!159760119038!e!!g!!elastic%20beanstalk&ef_id=WKgq9QAABVYkDTpR:20170222115859:s) (with a [self-signed SSL certificate](http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https-ssl.html)), or any other method you can think of.

We’re going to use [ngrok](https://ngrok.com/) to **Tunnel** to a local development server.

##### Setting up a Sinatra application

* Make the directory with mkdir hello\_world\_app
* Head into the directory with cd hello\_world\_app
* Set up a Ruby application with bundle init (you may need to install Bundler with gem install bundler first)
* Add the Sinatra gem to your Gemfile, by adding the line gem 'sinatra'
* Install Sinatra to your project using bundle install
* Create a server file with touch server.rb
* For now, create a single POST route, '/', that prints out the request body we are going to receive from Amazon:

# inside server.rb  
  
require 'sinatra'  
  
post '/' do  
 p request.body.read  
end

##### Opening your Sinatra application to the Internet using ngrok

* Download the appropriate ngrok package for your Operating System from the [ngrok downloads page](https://ngrok.com/download)
* Unzip the package and transfer the executable to your hello\_world\_app directory
* Start ngrok using ./ngrok http 4567
* Copy to the clipboard (command-C) the URL starting ‘https’ and ending ‘.ngrok.io’ from your ngrok Terminal
* In a second Terminal, start your Sinatra application using ruby server.rb.

## 3. Linking the Amazon VUI to our Backend, via the Endpoint

Our third step is to link the skill we set up on Amazon (1) with the Tunnel Endpoint (2) so our skill can send requests to our local application.

##### Configuring the Endpoint in the Alexa Skills Portal

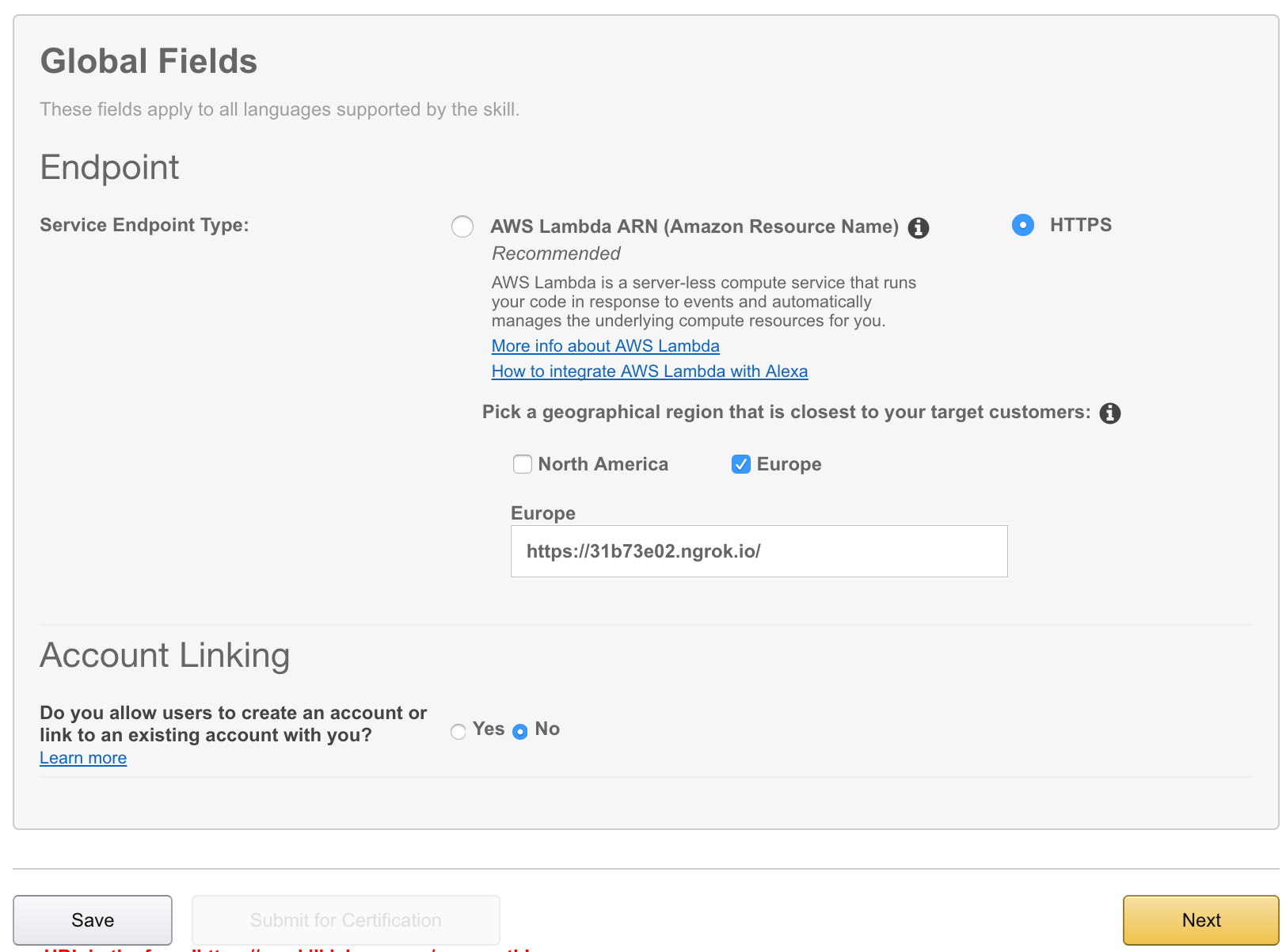
When Amazon invokes an Intent, Amazon sends a POST request to the specified *Endpoint* (web address).

Head back to your Alexa skill (for which you just entered Intents and Utterances). Hit Next, then set up the Endpoint.

* Use HTTPS, not Lambda (no Ruby on Lambda)
* Geographical Region: Europe
* Paste the Endpoint to your application into the text input field

If using ngrok, your Endpoint is the URL you copied, starting with ‘https’ and ending with ‘.ngrok.io’.

* You won't need Account Linking for this application.



##### Configuring SSL

Amazon Alexa only sends requests to secure Endpoints: ones secured using an SSL certificate (denoted by the 'S' in HTTPS). Since we used ngrok to set up our HTTPS Endpoint, we can use ngrok's 'wildcard' certificate instead of providing our own.

* If you used ngrok to set up a Tunnel, select ‘My development endpoint is a sub-domain of a domain that has a wildcard certificate from a certificate authority’.
* Hit 'Next' again.

##### Testing in the Service Simulator

The *Service Simulator* allows you to try out utterances. Once you’ve written an utterance into the Service Simulator, you can send test requests to the application endpoint you defined. You can see your application’s response to each request that you send.

* Use the Service Simulator to test that the say hello world utterance causes Amazon to send an Intent Request to your local application, and observe that the request body printed to the command-line matches the JSON request sent in the Service Simulator.

You’ve now hooked up your local development environment to an Alexa skill!

## 4. Responding to Alexa Requests

Now we have built an Alexa development skill (1), built a local development server with an endpoint tunnelled via HTTPS (2), and can make requests from Amazon to our local development server through that endpoint (3).

Our final step is to construct a response from our endpoint such that Amazon can interpret the response to make Alexa say ‘Hello World’ to us.

##### Building the JSON Response

Amazon sends and receives JSON responses in a particular format. Let's set that up here.

* require 'json' at the top of server.rb
* Replace the body of our Sinatra POST ‘/‘ route with the smallest possible response from the [Alexa Skills Kit Response Body Documentation](https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/alexa-skills-kit-interface-reference#response-body-syntax):

# inside server.rb  
  
post '/' do  
 {   
 version: "1.0",  
 response: {  
 outputSpeech: {  
 type: "PlainText",  
 text: "Hello World"  
 }  
 }  
 }.to\_json  
end

There are a few parts to this JSON response object:

* version (string): required. Allows you to version your responses.
* response (object): required. Tells Alexa how to respond: including speech, cards, and prompts for more information.
* outputSpeech (object). Tells Alexa what to say.
  + type (string) required. Tells Alexa to use Plain Text speech, where Alexa will guess pronunciation, or [Speech Synthesis Markup Language](https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/docs/speech-synthesis-markup-language-ssml-reference) (SSML), where you can specify pronunciation very tightly.
  + **EXTRA CREDIT**: Change the response to use custom pronunciation using SSML.
  + text (string) required. Tells Alexa exactly what to respond with.
  + **EXTRA CREDIT**: Play around with this response, restarting the server and sending an Intent Request from the Service Simulator each time.

##### Testing our Response in the Service Simulator...and beyond!

Now that we've built a JSON response, we can restart the server, and test out the new response in the Service Simulator.

If you would like to try your new Hello World skill out live, ask Hello World to say 'Hello World' on any Alexa-enabled device registered to your developer account!