DECOUPLED DRUPAL

With ReactJS and Alexa

Vincenzo Gambino

Senior Developer from Palermo, Italy.

12 years experience with Drupal

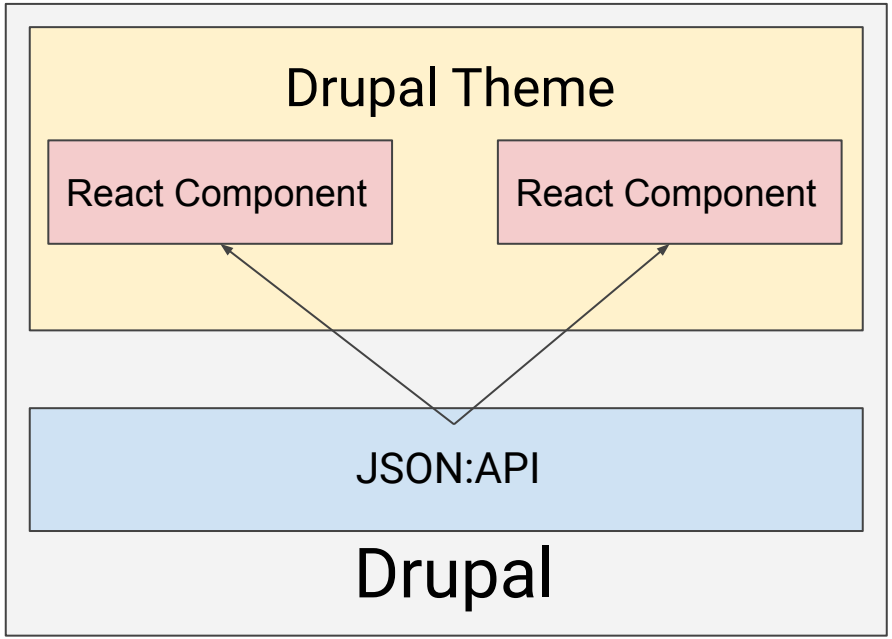
Speaker at DrupalCamp London, DrupalCamp Munich and Drupal Camp Poland.

Author of the Book: Jumpstart Jamstack Development: Build and deploy modern websites and web apps using Gatsby, Netlify, and Sanity.

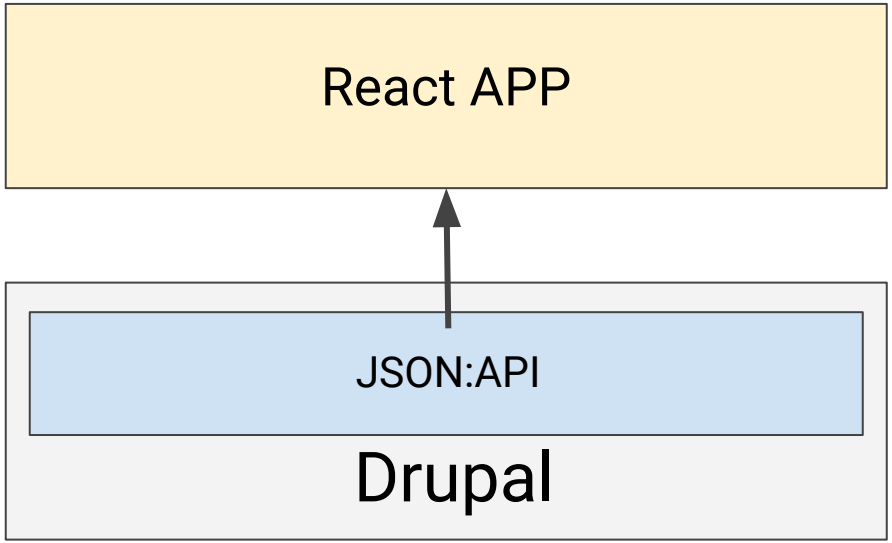
Agenda

- Progressive decoupled vs decoupled vs Headless
- React Intro
- Drupal with Embedded React
- Drupal and React APP
- Drupal and Alexa

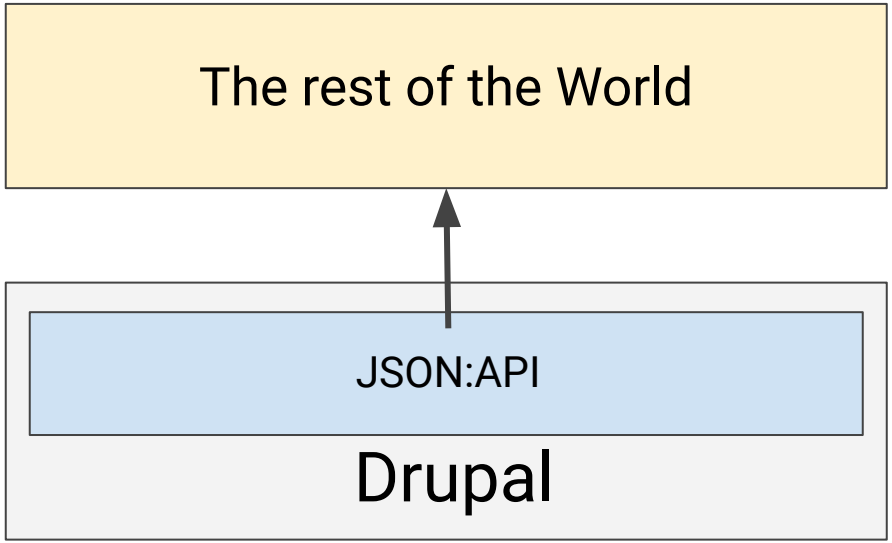
Progressive Decoupled



Decoupled



Headless



React Intro

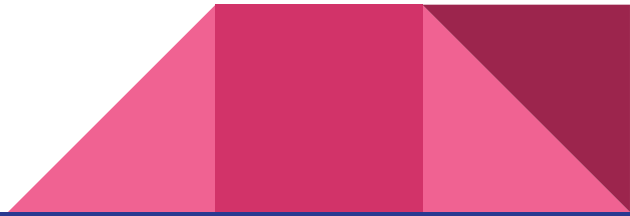
Components

- Components are the bricks of a ReactJS App.
- Components have their own logic
- Components can be reusable

JSX

- JSX is a syntax extension to JavaScript.
- It is possible to write HTML structure within the javascript file using JSX
- It is possible to write expressions

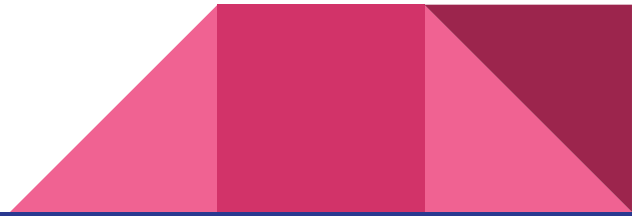
```
let name = ‘Vincenzo’;
const root = ReactDOM.createRoot(
  document.getElementById('root')
);
const element = <h1>Hello {name}</h1>;
root.render(element);
```



Drupal with Embedded React Components

Objectives

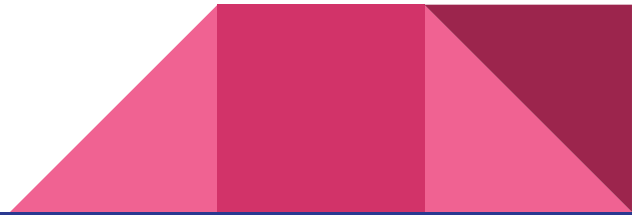
- Build a page with a component that retrieves Articles
- Add new node using a React
- Edit an existing node using React
- Use X-CSRF token to perform add and edit action



Theme setup - Install react packages

Inside your theme folder (themes/dmc_react) install the following packages:

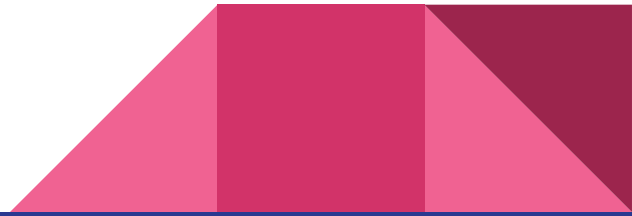
```
npm install --save react react-dom prop-types
npm install --save-dev @babel/core @babel/preset-env
@babel/preset-react babel-loader webpack webpack-cli
```



Configure Webpack

Inside your theme folder (themes/dmc_react) create the file webpack.config.js

```
const path = require('path');
const config = {
  entry: {
    main: ["./js/src/index.jsx"]
  },
  output: {
    path: path.resolve(__dirname, "js/dist"),
    filename: '[name].min.js'
  },
  resolve: {
    extensions: ['.js', '.jsx'],
  },
  module: {
    rules: [
      {
        test: /\.jsx?$/,
        loader: 'babel-loader',
        exclude: /node_modules/,
        include: path.join(__dirname, 'js/src'),
      }
    ],
  },};
module.exports = config;
```



Setup .babelrc

Inside your theme folder (themes/dmc_react) create the file .babelrc

```
{
  "presets": [
    "@babel/preset-env",
    "@babel/preset-react"
  ]
}
```

Setup the library

Inside your theme folder (themes/dmc_react) create the file dmc_react.libraries.yml

```
react_app:
  version: VERSION
  js:
    js/dist/main.min.js: {minified: true}
```

Attach the library

Inside your theme dmc_react.theme file attach the library.

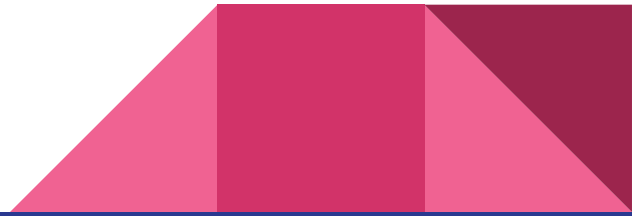
```
<?php
/**
 * Implements hook_page_attachments_alter().
 */
function dmc_react_page_attachments_alter(array &$attachments) {
    $attachments['#attached']['library'][$index] =
'dmc_react/react_app';
  }
}
```

Create the index.jsx

Create the folder src and inside, create the field index.jsx with the following content:

```
import { createRoot } from 'react-dom/client';
import React from "react";
const container = document.getElementById('dmc-react');
const root = createRoot(container);
root.render(<h1>Hello World!</h1>);
```

Run `npm run start` within your theme folder.



List, Add and Edit nodes

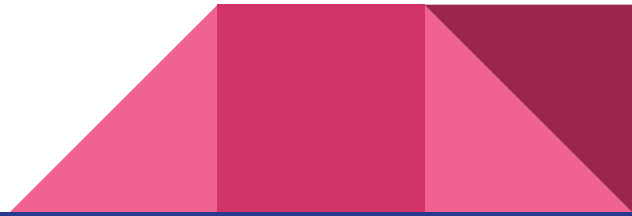
Modules

Enable the following modules

- JSON:API
- RESTful Web Services
- Serialization

Components involved

- ArticleListingPage
- NodeForm
- NodeAdd
- NodeEdit



ArticleListingPage component

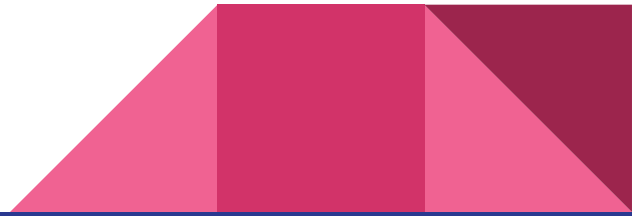
- Main wrapper for Article listing
- API call to JSON:API endpoint
- Includes components
  - NodeItem
  - NodeForm

NodeForm Component

Creates the Node Form with two fields:

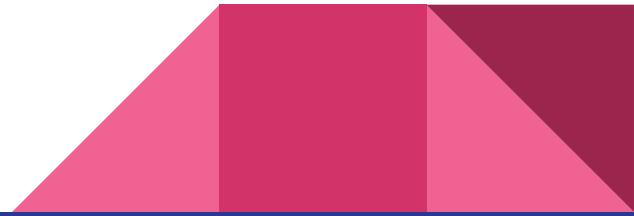
- Title
- Body

It will be used by both NodeAdd and NodeEdit component.



NodeAdd and NodeEdit Components

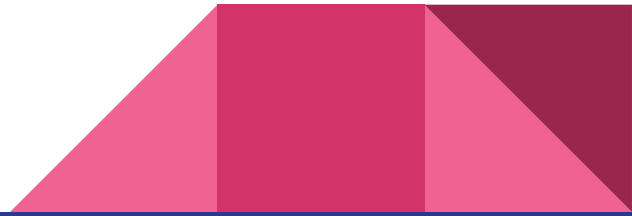
These two components will call NodeForm component but NodeAdd won’t pass any existing data while NodeEdit will pass existing data that will be shown in the form.



API Call example

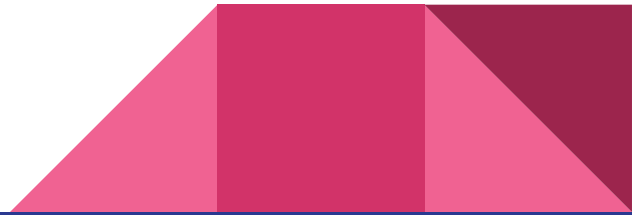
The API call will be made against the the JSON:API endpoint

In our case, we want to retrieve 10 articles, sort them by created date and we want only the id, the uuid, the title and the body.



Fetch call

```
const url =
`/jsonapi/node/article?fields[node--article]=id,drupal_internal__nid,title,body&
sort=-created&page[limit]=10`;

const headers = new Headers({
  Accept: 'application/vnd.api+json',
});

fetch(url, {headers})
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch(err => console.log(err));
```



Response

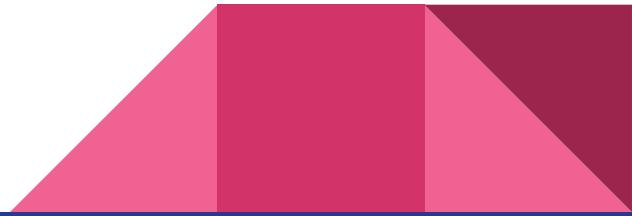

```
{
  "jsonapi": { ... },
  "data": [
    {
      "type": "node--article",
      "id": "UUID",
      "links": { … },
      "attributes": {
        "drupal_internal__nid": 1,
        "title": "Test for Drupal Mountain Camp",
        "body": {
          "value": "Lorem ipsum ...",
          "format": "plain_text",
          "processed": "<p>Lorem ipsum...",
          "summary": "Lorem ipsum ..."
        }
      }
    },
```

X-CSRF Token

For all the request that perform a state change, Drupal requires the X-CSRF token in the header for the request to avoid CSRF attacks.

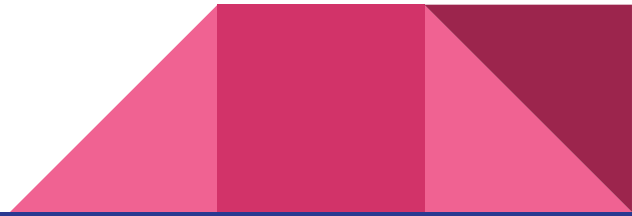
It can be retrieved from /session/token?_format=json making a fetch request.

The token can then be used to Add/Edit content.



POST Request

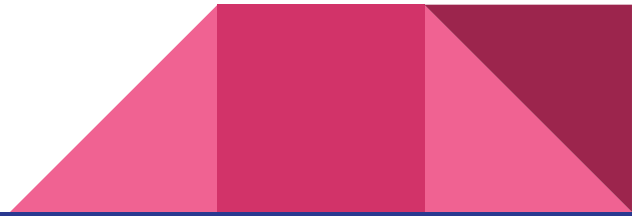
```
const headers = new Headers({
    'Accept': 'application/vnd.api+json',
  'Content-Type': 'application/vnd.api+json',
  'Cache': 'no-cache',
  'X-CSRF-Token': '{TOKEN}'
});
const options = {
  method: 'POST',
  credentials: 'same-origin',
  Headers,
  body: JSON.stringify(data)
}
fetch('/jsonapi/node/article', options).then(...);
```



Fully Decoupled React APP

Objectives

- Create a React APP
- Reuse the same component as the progressive decoupled app
- Use OAuth for authentication



Simple OAuth Drupal module

Download and install the simple_oauth drupal module and follow the installation instruction from https://www.drupal.org/project/simple_oauth

Create a new Client (Consumer) under admin/config/services/consumer

Fill the following fields:

- Label
- New Secret
- Scope

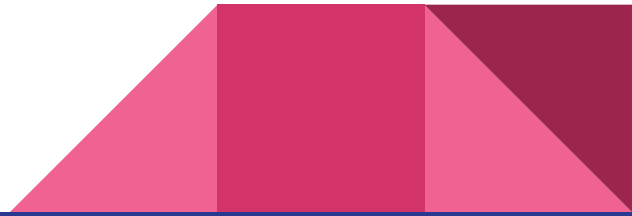
Drupal Enable CORS

To enable CORS in Drupal, open the file services.yml under sites/default folder. Is it’s not present, copy the file default.services.yml and rename it services.yml

```
  cors.config:
    enabled: true
    # Specify allowed headers, like 'x-allowed-header'.
    allowedHeaders: ['*']
    # Specify allowed request methods, specify ['*'] to allow all
possible ones.
    allowedMethods: ['*']
    # Configure requests allowed from specific origins.
    allowedOrigins: ['*']
```

React App component

- ArticleListingPage
- NodeForm
- NodeAdd
- NodeEdit
- Login

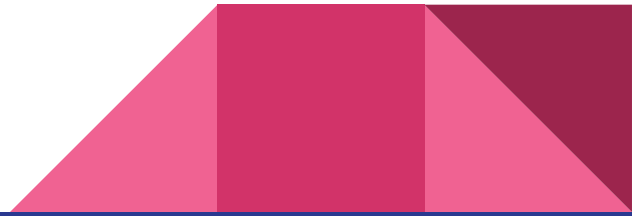


Login Component

This component contains a login form with Username and password field.

On submit it will make a request to the OAuth endpoint in drupal.

If the user is logged in, a welcome message will be shown instead.

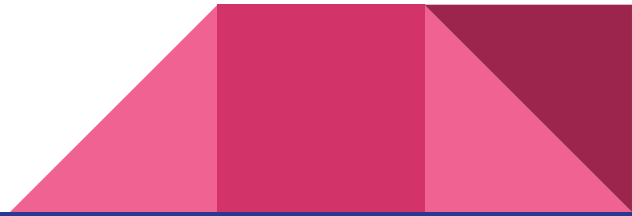


OAuth Request Headers and Params

```
var myHeaders = new Headers();
    myHeaders.append("Accept", "application/vnd.api+json");
    myHeaders.append("Content-Type",
"application/x-www-form-urlencoded");
    var urlencoded = new URLSearchParams();
    urlencoded.append("grant_type", "password");
    urlencoded.append("client_id", config.client_id);
    urlencoded.append("client_secret", config.client_secret);
    urlencoded.append("scope", config.scope);
    urlencoded.append("username", username);
    urlencoded.append("password", password);
```

OAuth fetch Request

```
    var requestOptions = {
      method: 'POST',
      headers: myHeaders,
      body: urlencoded,
      redirect: 'follow'
    };
     try {
       const response = await fetch(`${config.base}/oauth/token`,
        requestOptions
       );
    const data = await response.json();
```



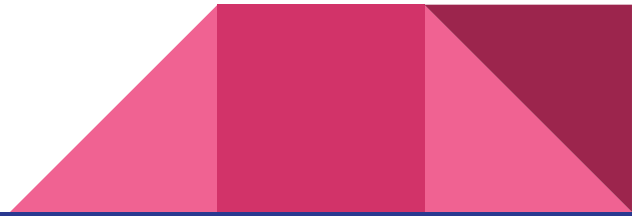
OAuth Reponse

The response will contain the following data:

```
{

    "token_type": "Bearer",
    "expires_in": 86400,
    "access_token": "...",
    "refresh_token": "..."

}
```

OAuth Save the token

```
    let token = Object.assign({}, data);
    token.date = Math.floor(Date.now() / 1000);
    token.expires_at = token.date + token.expires_in;
    localStorage.setItem('dmc-token', JSON.stringify(token));
```



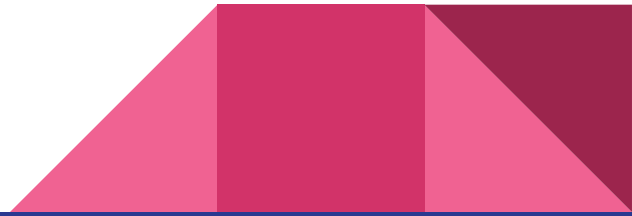
Post Request with Bearer Token

```
Const bearer =localStorage.getItem(‘dmc-token’);
const fetchOptions = {
method: 'POST',
      headers: new Headers({
        'Accept': 'application/vnd.api+json',
        'Content-Type': 'application/vnd.api+json',
        'Cache': 'no-cache',
        'Authorization': `Bearer ${bearer}`
      }),
      body: JSON.stringify(data),
    };
```

Alexa Skill

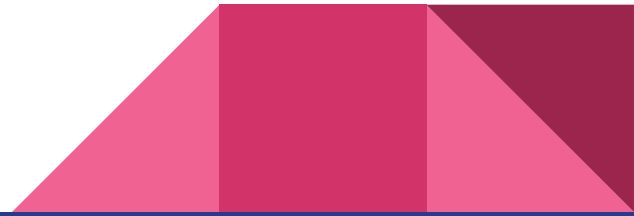
Objectives

- Create an Alexa Skill
- Create an Intent to retrieve the latest articles
- Create the Lambda function to request the content and send it back to the Alexa device.

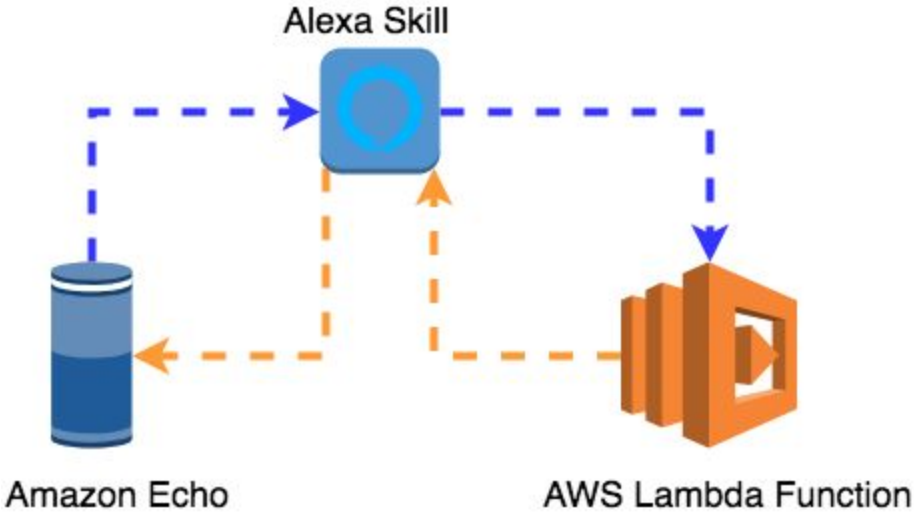


Alexa

Amazon Alexa is a virtual assistant developed by Amazon. It is capable of voice interaction, music playback, control several smart devices using itself as a home automation system. Users can extend the Alexa capabilities by installing "skills" (additional functionality developed by third-party vendors)



Alexa Skill Cycle



How to create an Alexa Skill

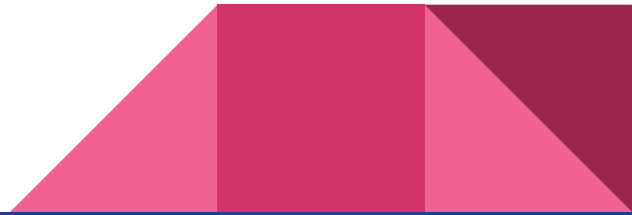
Invocation: Users say a skill's invocation name to begin an interaction with a particular custom skill.

Intents: An intent represents an action that fulfills a user's spoken request.

Slots: Slot types define how data in an intent slot is recognized and handled.

Interface: Interface components such as Audio player, Video app etc..

Endpoint: The Endpoint will receive JSON requests when a user interacts with your Alexa Skill.



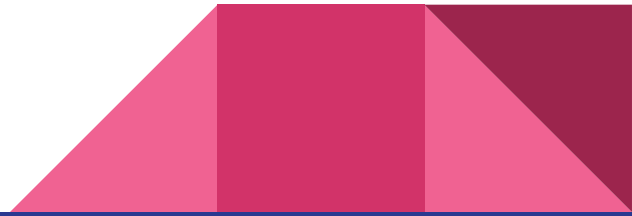
Drupal Mountain Camp Skill

Skill name: Drupal Mountain Camp

Language: English (UK)

Model: Custom

Skill’s backend: Alexa-hosted (node.js)

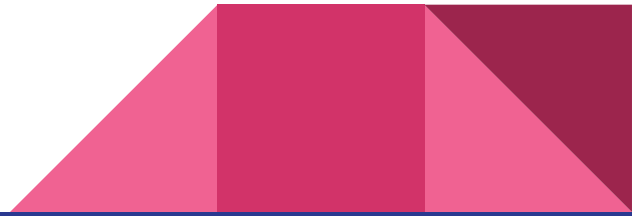


Invocation name

Users say a skill's invocation name to begin an interaction with a particular custom skill.

For example, if the invocation name is "drupal mountain camp", users can say:

User: Alexa, open drupal mountain camp



Intents, Utterances and Slots

An intent represents an action that fulfills a user's spoken request. Intents can optionally have arguments called slots. The Utterances map from a natural language query to an intent

Our skill has an intent named LatestArticlesIntent with the Utternace tell me the latest articles.

A user can then say:

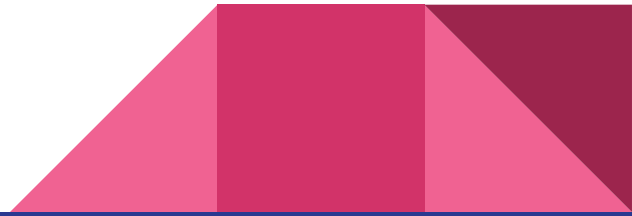
Alexa, ask drupal mountain camp to **tell me the latest articles**.

Endpoint

Here’s where you define custom code URL that implements your application.

It can point to a Lambda function or to a custom application endpoint.

Alexa Skill will send a JSON payload to your end point.

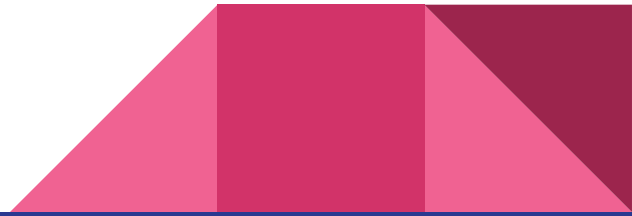


Intent: LatestArticlesIntent

Intent name: LatestArticlesIntent

Sample Utterance: tell me the latest articles

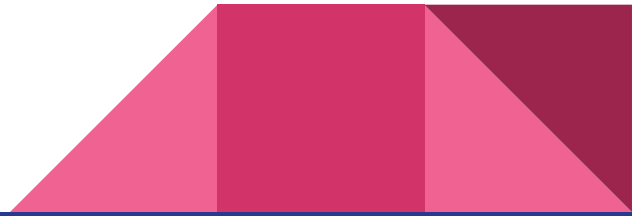
Alexa, ask drupal mountain camp to tell me the latest articles



Lambda function

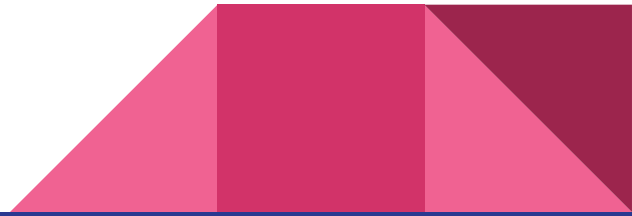
AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of the AWS. It runs code in response to events and automatically manages the computing resources required.

- Simplifies on-demand applications.
- Natively supports Java, Go, PowerShell, Node.js, C#, Python, and Ruby code.



IntentHandler Structure

```
const HelloWorldIntentHandler = {
    canHandle(handlerInput) {
     // Where we compare the RequestType and the IntentName
    },
    handle(handlerInput) {

    // Return the response.
    }
};
```



LatestArticlesIntentHandler - CanHandle

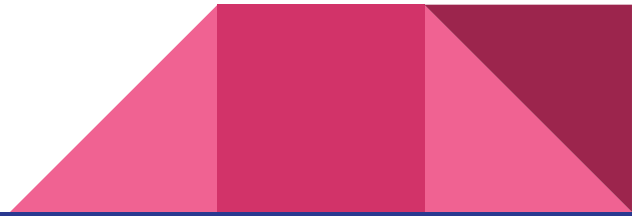
```
const LatestArticlesIntentHandler = {
    canHandle(handlerInput) {
    return Alexa.getRequestType(handlerInput.requestEnvelope)
=== 'IntentRequest'
        && Alexa.getIntentName(handlerInput.requestEnvelope)
=== 'LatestArticlesIntent';
    },
```

LatestArticlesIntentHandler - Handle

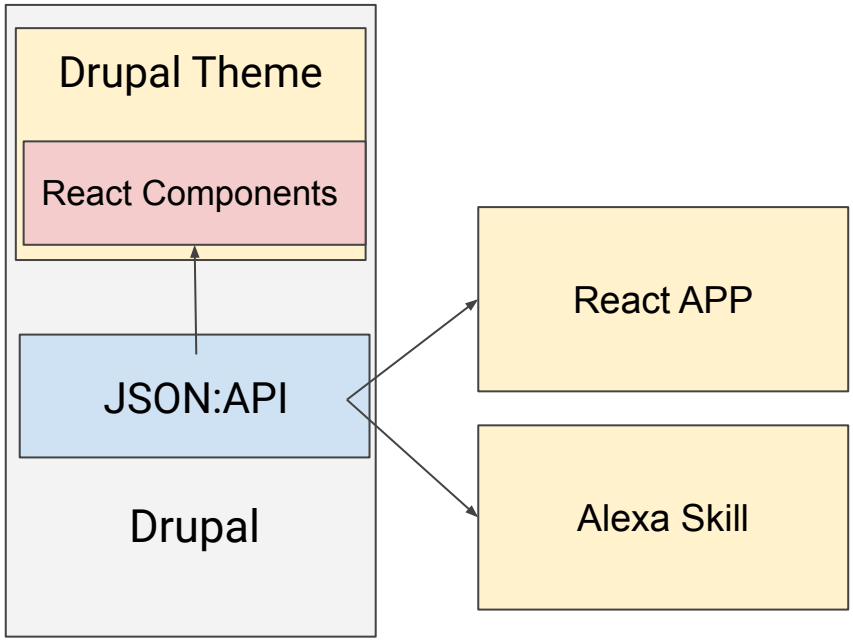
```
        async handle(handlerInput) {
             const articleTitles = await getArticles();
             let len = articleTitles.length;
             let speakOutput = '';
             if (len === 0) {
                 speakOutput = 'There are no articles.';
             }
             else {
                  speakOutput = `I have found ${len} articles. ${articleTitles.join(', ')}`;
             }
             handlerInput.responseBuilder
                 .speak(speakOutput)
              return handlerInput.responseBuilder

             .getResponse();

         }
```

LatestArticlesIntentHandler - GetArtciles Request

```
var options = {
  'method': 'GET',
  'hostname': 'dev-dmc-react.pantheonsite.io',
  'path': '/api/articles',
  'headers': {
    'Accept': 'application/vnd.api+json'
  },
  'maxRedirects': 20
};
var req = https.request(options, function (res) {});
```



Final Structure



Resources

Pantheon: https://pantheon.io/

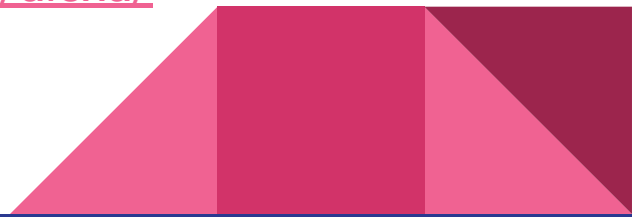
React: https://reactjs.org/

Drupal and React: https://reactfordrupal.com/

ContentaCMS: https://www.contentacms.org/

Amazon Developer Console: https://developer.amazon.com/

Alexa Developer Console: https://developer.amazon.com/alexa/



Honorable Mentions

Drupalize.me course: Introduction to React and Drupal

Base code: https://github.com/DrupalizeMe/react-and-drupal-examples/

Q&A