

microCMS+Next.jsでブログ作成

チュートリアルをやってみた



なぜこのチュートリアル?

- microCMS+Next.jsな案件がくる(かも)
- 最近フロントエンド関連の勉強会参加していた
- 教材
 - [microCMS + Next.jsでJamstackブログを作ってみよう | microCMSブログ](#)

やったこと

- Next.jsでブログ用コード準備
- **microCMS**でAPI作成
- VercelにリポジトリとWebhook連携



microCMSについて教えてください



microCMSとは、Webアプリケーションを構築するためのCMS（Content Management System）です。microCMSは、業界を問わず、Webアプリケーションを構築するために必要な機能を提供します。そのため、microCMSは、Webサイトの構築やコンテンツの管理を行うための最適なツールとして広く活用されています。



microCMSは、簡単にWebアプリケーションを構築することができるため、Web制作会社やフリーランスのWebデザイナーなどが、専門的な知識を必要とせずにWebアプリケーションを構築することができます。また、microCMSは、サービスの開発・運用を担う開発者にとっても、設計や開発のプロセスを効率化するための有用なツールとなっています。

microCMSには、さまざまな機能が用意されており、それらを組み合わせることで、様々なWebアプリケーションを構築することができます。たとえば、microCMSには、ユーザー管理やページの管理、投稿記事の管理、メニューの管理、データベースの管理、その他多くの機能があります。また、microCMSには、Webアプリケーションを構築するためのAPI（Application Programming Interface）が用意されており、それらを活用することで、より複雑なWebアプリケーションを構築す

Next.js

- ブログ用コード準備
- `npx create-next-app microcms-next-jamstack-blog`
 - 最新版の13で生成される
- microCMSのSDKをinstall
 - ↑経由でAPIアクセス
- ハマりどころ
 - `<a>`タグで囲むとエラーなので除去した
 - Next13からLinkタグの書き方変わった?

microCMS

- API作成
- Vercelを連携
 - Webhook利用

The screenshot displays the 'APIスキーマ' (API Schema) configuration page in microCMS. The interface is divided into several sections:

- API設定 (API Settings):** Includes '基本情報' (Basic Information), 'APIスキーマ' (API Schema), 'カスタムステータス' (Custom Status), 'コンテンツID' (Content ID), '画面プレビュー' (Preview), 'Webhook', and 'IP制限' (IP Restriction).
- ドキュメント (Documentation):** Includes 'ドキュメント' (Documentation) and '重要な操作' (Important Operations).
- APIスキーマ (API Schema):** The main configuration area showing three fields:
 - QLMrrmeg7B:** Field ID 'title', Display Name 'タイトル', Type 'テキストフィー...' (Text Field), and '必須項目' (Required) toggle.
 - p10vQuPcYt:** Field ID 'body', Display Name '本文' (Body), Type 'リッチエディタ' (Rich Editor), and '必須項目' (Required) toggle.
 - cz_rXsCnU5:** Field ID 'category', Display Name 'カテゴリー' (Category), Type 'コンテンツ参照...' (Content Reference), and '必須項目' (Required) toggle.

At the bottom, there is a '+ フィールドを追加' (Add Field) button and a '変更する' (Change) button. A link 'この設定をエクスポートする' (Export this setting) is also present.

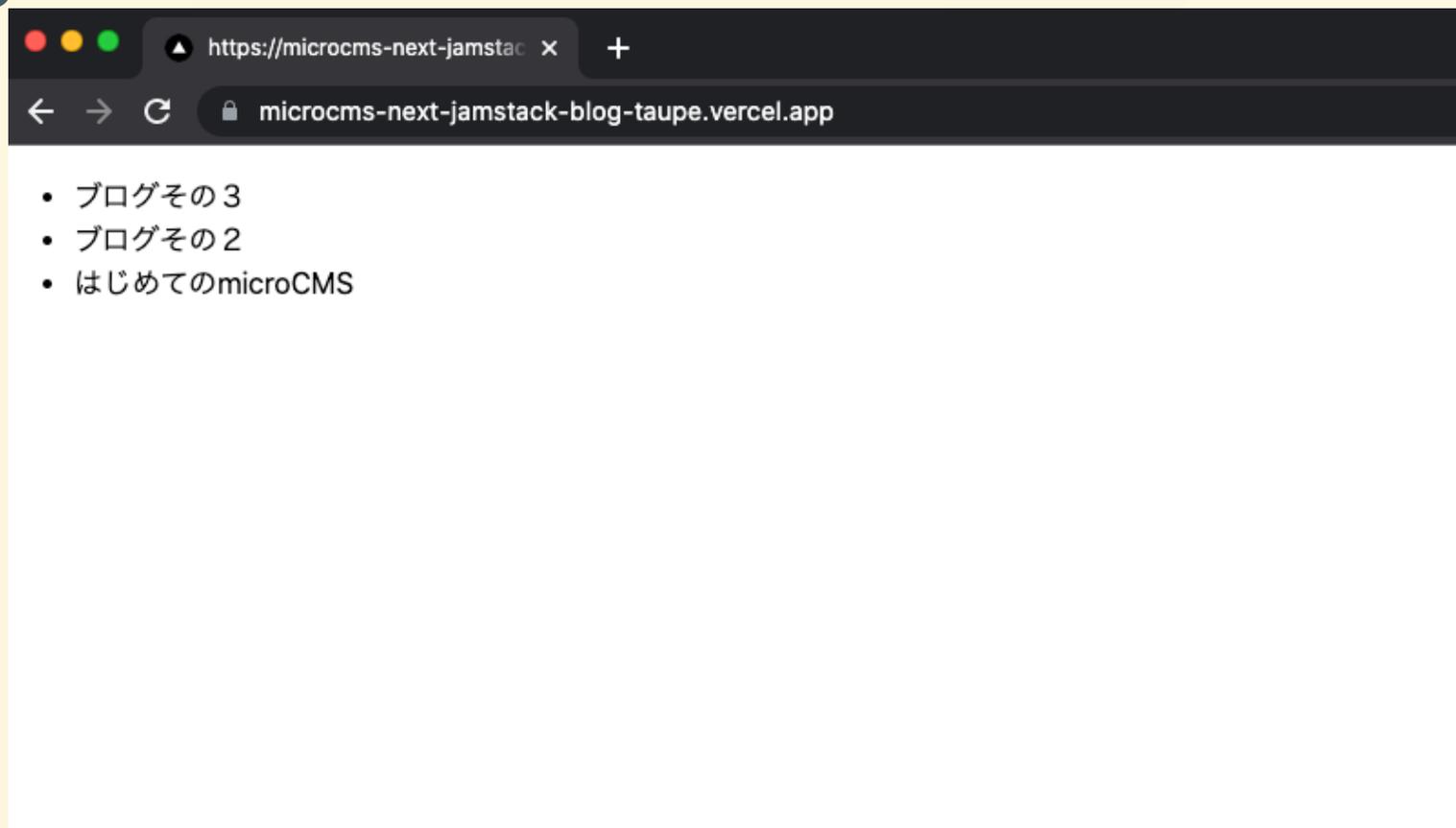
Vercel

- CI/CDとWebサーバーが合わさったサービス
- トリガー設定
 - Github連携
 - microCMSとのWebhook設定

The screenshot shows the Vercel deployment page for a project named 'microcms-next-jamstack-blog'. The interface includes a navigation bar with 'Deployment', 'Functions', and 'Source' tabs. The main content area displays deployment details for a 'Production' environment, which is 'Ready' and took '23s (4d ago)' to complete. The domain is 'microcms-next-jamstack-blog-taupe.vercel.app'. The current branch is 'main' with commit '336c9f6'. A red box highlights the 'main' branch, with a red arrow pointing to it and the text 'Mainブランチ' below. Below this, the 'Deployment Status' section shows a 'Building' phase with a log of commands and their execution times. A red box highlights the log entry 'Running "npm run build"', with a red arrow pointing to it and the text 'npm run build' above it.

```
16:16:04.088 Cloning github.com/taichiyam/microcms-next-jamstack-blog (Branch: main, Commit: 336c9f6)
16:16:06.008 Cloning completed: 1.920s
16:16:07.192 Restored build cache
16:16:07.249 Running "vercel build"
16:16:07.922 Vercel CLI 28.6.0
16:16:08.393 Installing dependencies...
16:16:09.537 up to date in 778ms
16:16:09.538 88 packages are looking for funding
16:16:09.538   run `npm fund` for details
16:16:09.549 Detected Next.js version: 13.0.5
16:16:09.553 Detected `package-lock.json` generated by npm 7+...
16:16:09.553 Running "npm run build"
16:16:09.962 > microcms-next-jamstack-blog@0.1.0 build
16:16:09.963 > next build
16:16:09.963 info - Linting and checking validity of types...
```

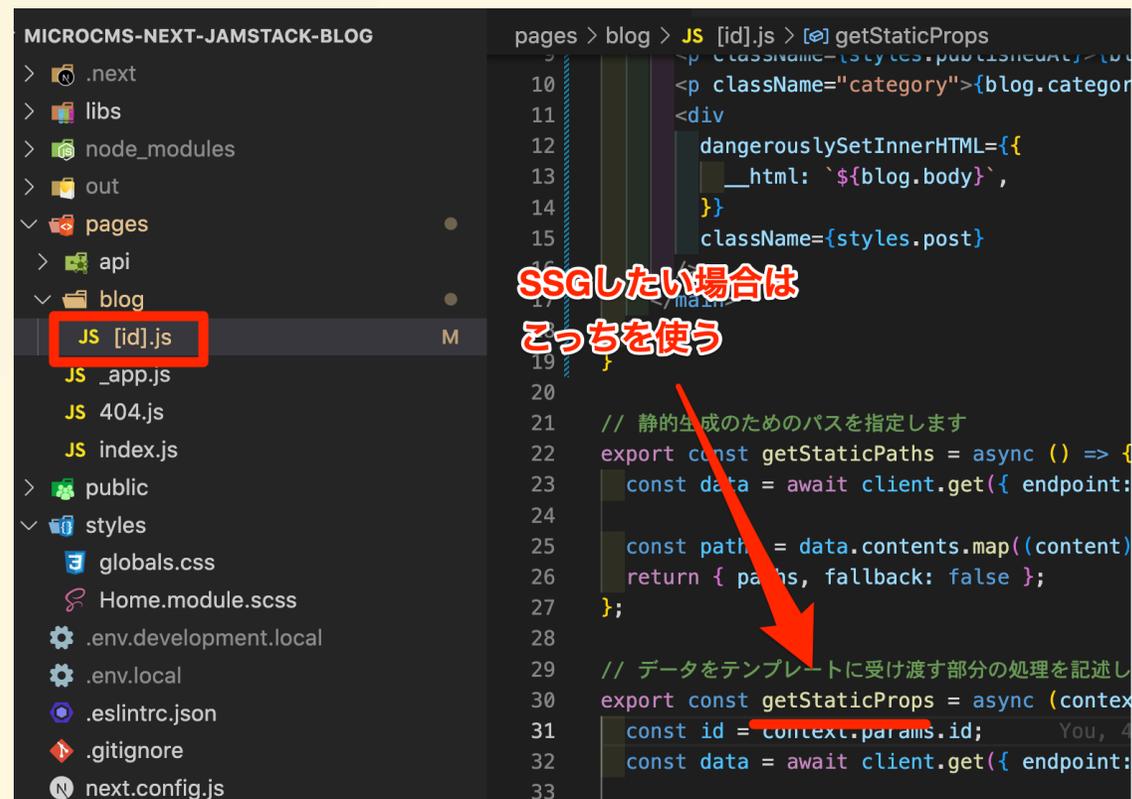
成果物



わかったこと

Next.js

- (コピペなのでコードはわからん)
- ルーティング書くんじゃなくて配置場所で
 - ファイルベースルーティング
 - myblog.com/blog/12
 - `blog/[id].js`



```
MICROCMS-NEXT-JAMSTACK-BLOG
> .next
> libs
> node_modules
> out
> pages
> api
> blog
  JS [id].js
  JS _app.js
  JS 404.js
  JS index.js
> public
> styles
  globals.css
  Home.module.scss
.env.development.local
.env.local
.eslintrc.json
.gitignore
next.config.js

pages > blog > JS [id].js > getStaticProps
10 <p className={styles.publishedDate}>{blog.publishedDate}</p>
11 <p className="category">{blog.category}</p>
12 <div
13   dangerouslySetInnerHTML={{
14     __html: `${blog.body}`,
15   }}
16   className={styles.post}
17 >{blog.body}</div>
18 }
19 }
20
21 // 静的生成のためのパスを指定します
22 export const getStaticPaths = async () => {
23   const data = await client.get({ endpoint: 'posts' });
24
25   const paths = data.contents.map((content) => {
26     return { params: { id: content.id }, fallback: false };
27   });
28
29 // データをテンプレートに受け渡す部分の処理を記述し
30 export const getStaticProps = async (context) => {
31   const id = context.params.id;
32   const data = await client.get({ endpoint: 'posts', params: { id } });
33 }
```

SSGしたい場合は
こっちを使う

microCMS

- APIスキーマ設定は(割とすぐ)習得できそう
- APIにv1とかついてるの良い
- スキーマやデータのimport/exportできる
- 実運用考えると?
 - 本番/検証/開発準備したいならビジネスプラン
 - ローカル開発時は自前でAPIサーバ建てる?
 - プレビューは?
 - これも自前で用意する必要がある

Vercel

- トリガーでビルド可能
 - ブランチのpush
 - 記事更新
- 無料枠多め

次にやること

- React
- Nextチュートリアルやる
 - Vercel社員(日本人)作成らしい

Appendix

APIのmock

Mockoon

[MockoonでREST APIのモック環境を爆速で立ち上げてみた](#)
[外部連携APIのMock API Server化を検討する \(Mockoon\) - Qiita](#)
[シンプルで使いやすい REST API Mockサーバ Mockoon](#)

json-server

[json-serverで手軽にREST APIのモックサーバーを作る](#)