

結局Nuxt Layersって 何ができて何が嬉しいの？

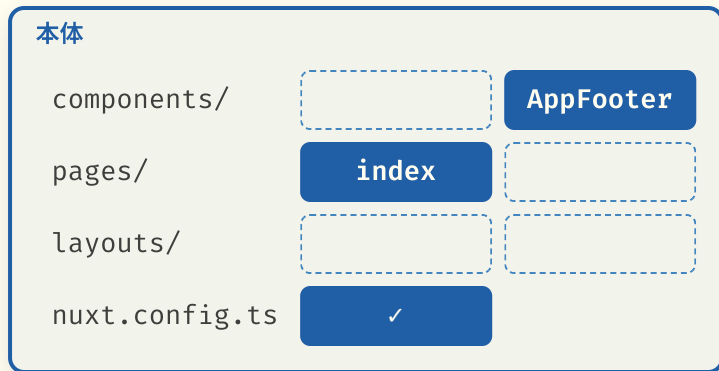
ryuhei373 / 株式会社ノーススター

Vue Fes Japan 2026

Nuxt Layers、知ってますか？

透明な型紙の例え

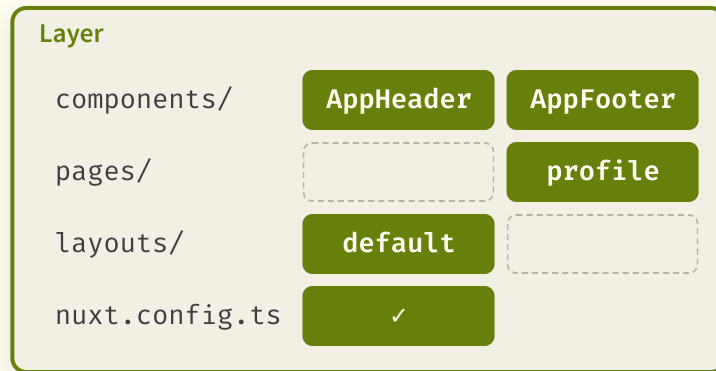
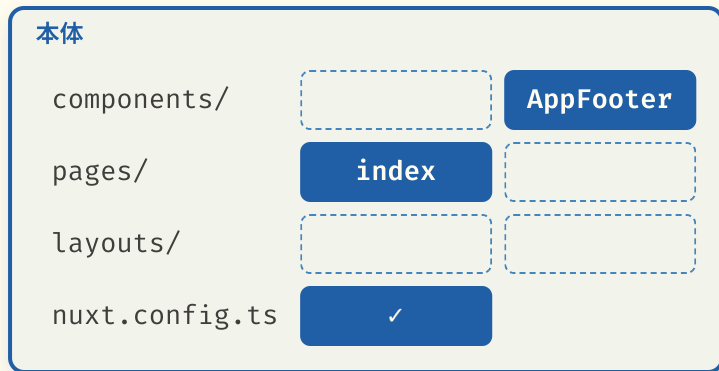
Nuxtアプリ（本体） → この型紙の枠にファイルを置くと、Nuxtが走査して組み立てる



透明な型紙の例え

Nuxtアプリ（本体）→ この型紙の枠にファイルを置くと、Nuxtが走査して組み立てる

Layer → 同じ型紙のファイル群。単体で起動して画面にならないものでも良い。（今回、破片と呼ぶ）

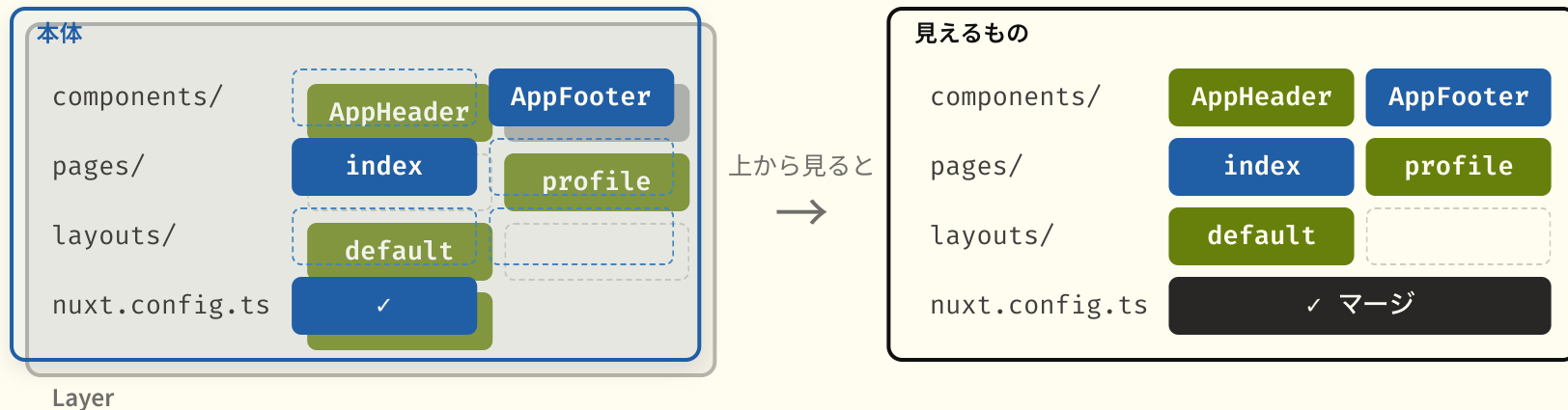


透明な型紙の例え

Nuxtアプリ（本体）→ この型紙の枠にファイルを置くと、Nuxtが走査して組み立てる

Layer → 同じ型紙のファイル群。単体で起動して画面にならないものでも良い。（今回、破片と呼ぶ）

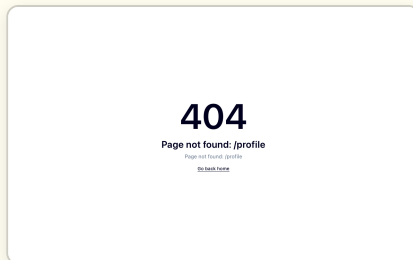
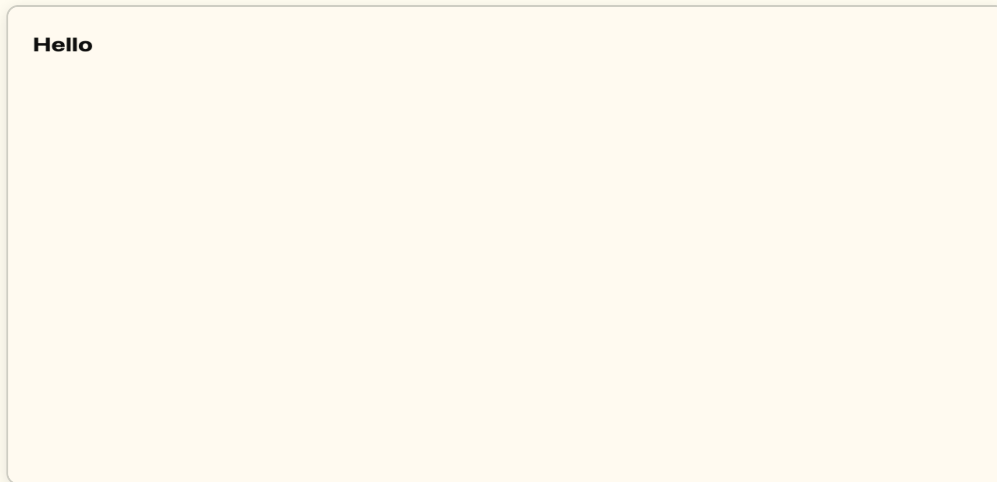
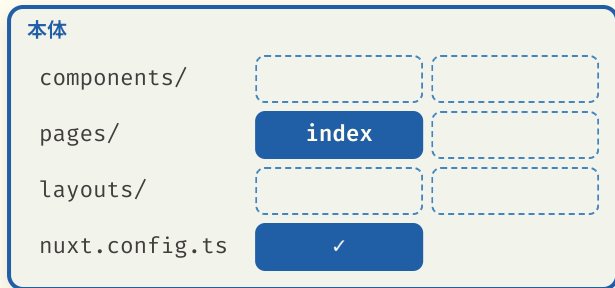
重ねると、上に無いものだけが下から透けて見える nuxt.config は隠れず、両方の中身がマージされる



「Layer」という名前からは上に重なっていく印象を受けるが、本体の下に敷いていくイメージを持つとわかりやすい。本体はいつも一番上

デモ Before

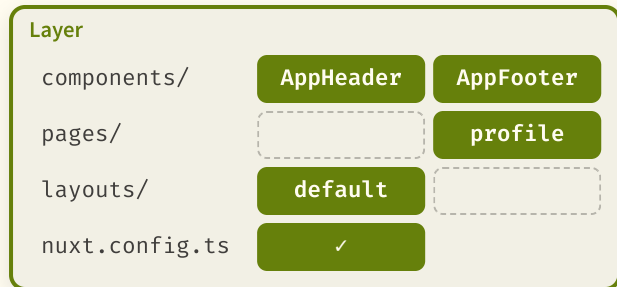
本体だけ。 `/profile` は無い



`/profile` は 404

敷く破片: base-layer/

ヘッダー・フッター・layout・プロフィールページだけを持つ



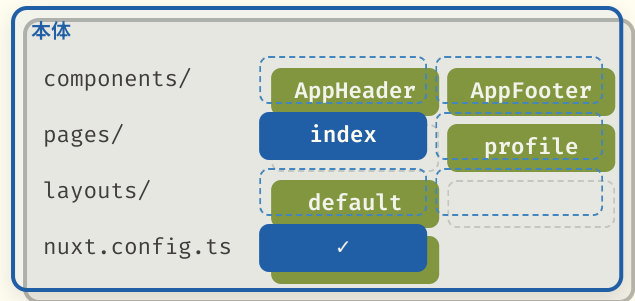
```
base-layer/  
├── app/  
│   ├── components/  
│   │   ├── AppHeader.vue  
│   │   └── AppFooter.vue  
│   ├── layouts/  
│   │   └── default.vue  
│   └── pages/  
│       └── profile.vue  
└── nuxt.config.ts ← siteName: 'Demo Site'
```

app.vue も pages/index.vue も無い。単体で起動しても画面にならない Nuxt アプリの破片

デモ After: 破片を敷く

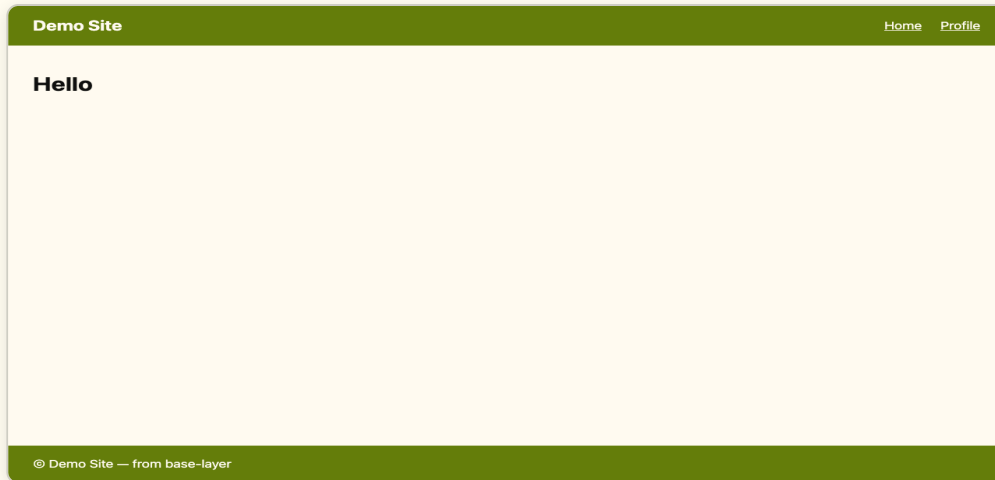
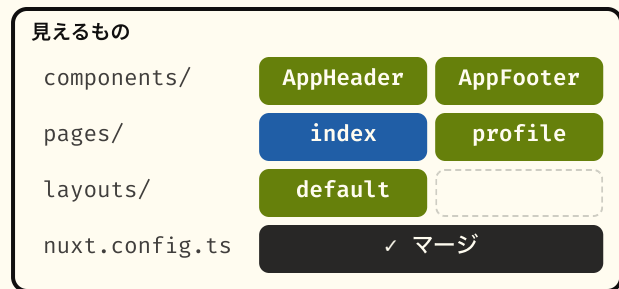
本体の nuxt.config.ts に1行

```
+extends: ['./base-layer'],
```



Layer

上から見ると ↓



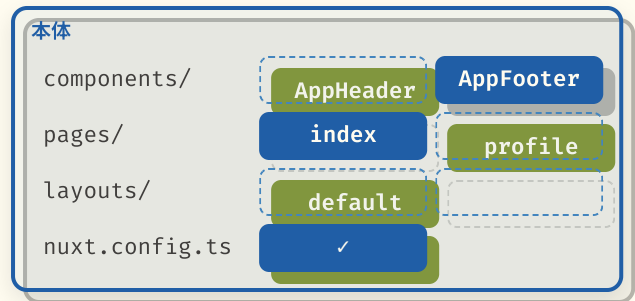
/profile も開く

ヘッダーの「Demo Site」は Layer の nuxt.config から。本体の config には書いていない

```
// base-layer/nuxt.config.ts
runtimeConfig: {
  public: { siteName: 'Demo Site' },
},
```

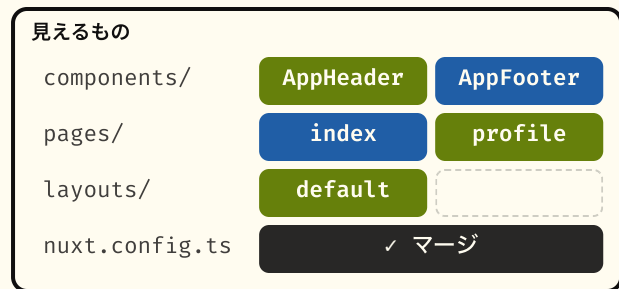
デモ 上書き: 本体が勝つ

本体に `components/AppFooter.vue` を1つ追加



Layer

上から見ると ↓



何ができる、何が嬉しいか

何ができる、何が嬉しいか

1. Nuxt と同じ構造で機能を切り出せる（パッケージにしなくていい）

→ 1つのアプリケーションとして成立していなくても、デザインや機能を同じ構造で共有できる

何ができ、何が嬉しいか

1. Nuxt と同じ構造で機能を切り出せる（パッケージにしなくていい）

→ 1つのアプリケーションとして成立していなくても、デザインや機能を同じ構造で共有できる

2. 置き場所を問わない（同じリポジトリ / リモートリポジトリ / レジストリ）

→ 同じリポジトリ内での切り出しから始め、完全に別リポジトリにしてもいいし、もちろんレジストリに公開してもいい。利用側は `extends` を1行直すだけ

何ができ、何が嬉しいか

1. Nuxt と同じ構造で機能を切り出せる（パッケージにしなくていい）

→ 1つのアプリケーションとして成立していなくても、デザインや機能を同じ構造で共有できる

2. 置き場所を問わない（同じリポジトリ / リモートリポジトリ / レジストリ）

→ 同じリポジトリ内での切り出しから始め、完全に別リポジトリにしてもいいし、もちろんレジストリに公開してもいい。利用側は `extends` を1行直すだけ

3. 同名のファイルは利用側が優先される

→ Layer側の肥大化を心配せずに、利用側で好きな部分を好きなように差し替えられる。

何ができ、何が嬉しいか

1. Nuxt と同じ構造で機能を切り出せる（パッケージにしなくていい）

→ 1つのアプリケーションとして成立していなくても、デザインや機能を同じ構造で共有できる

2. 置き場所を問わない（同じリポジトリ / リモートリポジトリ / レジストリ）

→ 同リポジトリ内での切り出しから始め、完全に別リポジトリにしまってもいいし、もちろんレジストリに公開してもいい。利用側は `extends` を1行直すだけ

3. 同名のファイルは利用側が優先される

→ Layer側の肥大化を心配せずに、利用側で好きな部分を好きなように差し替えられる。

4. `nuxt.config / app.config` はマージされる

→ 設定ごと配れる。`extends` 1行で「いつもの構成」から始まり、設定のずれがうっかり起きない

利用例 1: 機能ごと敷く

```
auth-layer/  
├── app/  
│   ├── middleware/  
│   │   └── auth.global.ts    ← 全ページにログインガード  
│   ├── pages/  
│   │   └── login.vue  
│   ├── composables/  
│   │   └── useAuth.ts  
└── nuxt.config.ts
```

```
// 本体 nuxt.config.ts  
extends: ['./auth-layer']
```

- middleware も pages も composable も、1つの破片にまとめて1行で付く
- extends を条件で切り替えれば、staging だけ敷く、のような付け外しもできる

利用例 2: テーマを配布する

```
theme-layer/  
├── app/  
│   ├── assets/  
│   │   └── theme.css      ← 配色、ダークモード  
│   └── app.config.ts      ← ui.colors など  
└── nuxt.config.ts        ← css の登録
```

```
// 本体 nuxt.config.ts  
extends: ['github:ryuhei373/nuxt-layer-flexoki#v0.2.0']
```

- CSS と app.config だけ。コンポーネントは1つも無い
- 自分のブログから切り出して、別プロジェクトで使い回している

github.com/ryuhei373/nuxt-layer-flexoki

利用例 3: ドメイン単位で分割する

```
my-site/  
├─ nuxt.config.ts  
├─ core/      ← middleware / plugins / utils  
├─ site/      ← 横断的な assets、about ページ  
├─ blog/      ← blog の pages / components / content  
└─ shop/      ← shop の pages / components / stores
```

```
// nuxt.config.ts は extends だけ  
extends: ['./core', './site', './blog', './shop']
```

- ファイルの種類 (components / pages) ではなく、機能の領域 (blog / shop) で切る
- 本体はほぼ空になり、アプリは「破片の集まり」になる

Dave Stewart, "Modular site architecture with Nuxt layers" (2024) — davestewart.co.uk/blog/nuxt-layers/