一、Gridsome 介绍

为什么

是什么

- GitHub 仓库：https://github.com/gridsome/gridsome
- 官网：https://gridsome.org/
- Gridsome 是由Vue.js驱动的Jamstack框架，用于构建默认情况下快速生成的静态生成的网站和应用。
- Gridsome是Vue提供支持的静态站点生成器，用于为任何无头CMS，本地文件或API构建可用于CDN的网站
- 使用Vue.js，webpack和Node.js等现代工具构建网站。通过npm进行热重载并访问任何软件包，并使用自动前缀在您喜欢的预处理器（如Sass或Less）中编写CSS。
- 基于 Vue.js 的 Jamstack 框架
- Gridsome 使开发人员可以轻松构建默认情况下快速生成的静态生成的网站和应用程序
- Gridsome允许在内容里面引用任何CMS或数据源。
  从WordPress，Contentful或任何其他无头CMS或API中提取数据，并在组件和页面中使用GraphQL访问它。

为什么选择 Gridsome

- **Vue.js for frontend** - The simplest & most approachable frontend framework.
- **Data sourcing** - Use any Headless CMSs, APIs or Markdown-files for data.
- **Local development with hot-reloading** - See code changes in real-time.
- **File-based page routing** - Any `Name.vue` file in `src/pages` is a static route.
- **Dynamic routing** - Any `[param].vue` file in `src/pages` is a dynamic route.
- **Static file generation** - Deploy securely to any CDN or static web host.
- **GraphQL data layer** - Simpler data management with a centralized data layer.
- **Automatic Code Splitting** - Builds ultra performance into every page.
- **Plugin ecosystem** - Find a plugin for any job.

什么是 Jamstack

Gridsome是一个Jamstack框架。 Jamstack使您可以通过预渲染文件并直接从CDN直接提供文件来构建快速安全的站点和应用程序，而无需管理或运行Web服务器。

Learn more about the Jamstack.

它是如何工作的

Gridsome生成静态HTML，一旦加载到浏览器中，该HTML就会渗入Vue SPA。这意味着您可以使用Gridsome构建静态网站和动态应用程序。

Gridsome为每个页面构建一个.html文件和一个.json文件。加载第一页后，它仅使用.json文件来预取和加载下一页的数据。它还为需要它的每个页面构建一个.js包（代码拆分）。

它使用vue-router进行SPA路由，并使用vue-meta来管理。

Gridsome默认添加最小57kB的gzip JS捆绑包大小（vue.js，vue-router，vue-meta和一些用于图像延迟加载的文件）。

详细了解其工作原理

学习条件

您应该具有有关HTML，CSS，Vue.js以及如何使用终端的基本知识。了解GraphQL的工作原理是有好处的，但不是必需的。Gridsome是学习它的好方法。

Gridsome 需要Node.js（v8.3 +），并建议使用 Yarn。

备选方案

- VuePress
- Nuxt
- Gatsby.js

使用场景

- 不适合管理系统
- 简单页面展示
- 想要有更好的 SEO
- 想要有更好的渲染性能

二、起步

> 目标：快速了解 Gridsome 项目

1、安装 Gridsome CLI

```
# 使用 yarn
yarn global add @gridsome/cli

# 使用 npm
npm install --global @gridsome/cli


# 查看是否安装成功
gridsome --version
```

2、创建 Gridsome 项目

```
# 创建项目
gridsome create my-gridsome-site

# 进入项目中
cd my-gridsome-site

# 启动开发模式，或 npm run develop
gridsome develop
```

> gridsome 项目安装依赖注意事项：
>
> - 配置 node-gyp 编译环境
>   - https://github.com/nodejs/node-gyp

> - 配置环境变量： `npm_config_sharp_libvips_binary_host` 为 `https://npm.taobao.org/mirrors/sharp-libvips/`
>   - https://github.com/lovell/sharp-libvips
>   - https://developer.aliyun.com/mirror/NPM
>   - https://npm.taobao.org/mirrors
>   - https://sharp.pixelplumbing.com/install
>   - `npm config set sharp_binary_host "https://npm.taobao.org/mirrors/sharp"`
>   - `npm config set sharp_libvips_binary_host "https://npm.taobao.org/mirrors/sharp-libvips"`
> - 配置 hosts： `199.232.68.133 raw.githubusercontent.com`
>   - https://www.ipaddress.com/

3、目录结构

```
.
├── src
│   ├── components # 公共组件
│   ├── layouts # 布局组件
│   ├── pages # 页面路由组件
│   ├── templates # 模板文件
│   ├── favicon.png # 网站图标
│   └── main.js # 应用入口
├── static # 静态资源存储目录，该目录中的资源不做构建处理
├── README.md
├── gridsome.config.js # 应用配置文件
├── gridsome.server.js # 针对服务端的配置文件
├── package-lock.json
└── package.json
```

4、自己试一试

- 在 `src/pages` 目录中创建一个 `.vue` 组件

5、构建

```
gridsome build
```

构建结果默认输出到 `dist` 目录中。

Gridsome 会把每个路由文件构建为独立的 HTML 页面。

6、部署

可以把构建结果 `dist` 放到任何 Web 服务器中进行部署。

例如我们这里使用 Node.js 命令行工具 serve 来测试构建结果。

```
npm install -g serve

serve dist
```

或者可以部署到其它第三方托管平台：https://gridsome.org/docs/deployment/。

或是自己的服务器，都可以！

核心概念

> 目标：学习 Gridsome 的核心概念

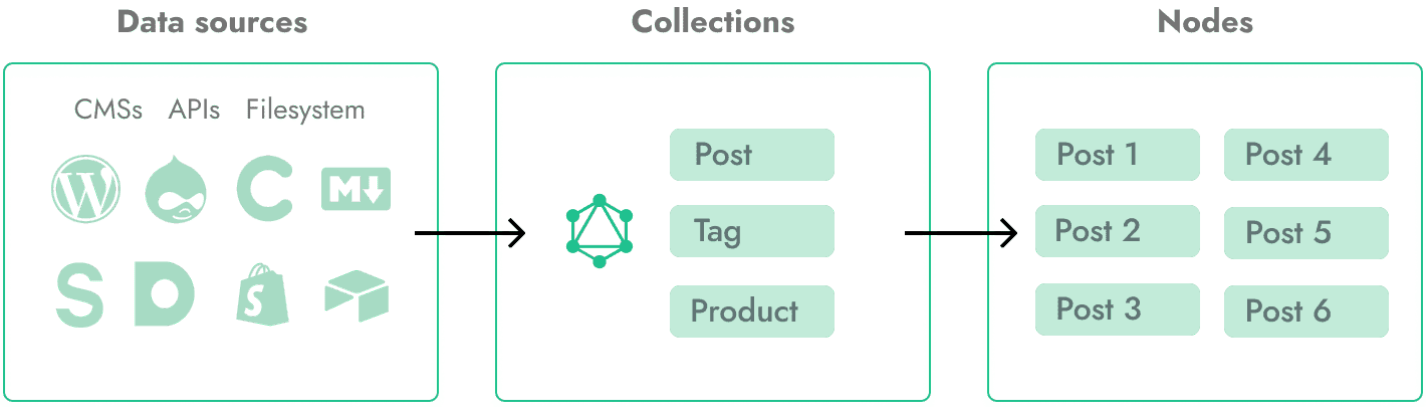
Pages

通过在 src/pages 文件夹中添加Vue组件来创建页面。他们使用基于文件的路由系统。例如，src / pages / About.vue将是mywebsite.com/about/。页面用于简单页面和列出集合的页面（例如/ blog /）。

了解有关页面的更多信息：https://gridsome.org/docs/pages/。

Collections

如果您要在网站上放置博客文章，标签，产品等，则收藏很有用。可以使用 Source插件或 Data Store API 从任何Headless CMS，内容API或Markdown文件中获取集合。



集合存储在临时的本地GraphQL数据层中，可以在任何地方查询，过滤，分页或有关系。

Templates

模板负责显示集合的节点（单个页面）。模板通常位于src / templates中。如果未在模板配置中指定组件，则Gridsome尝试查找与集合名称相同的文件。

这是一个例子：

```
<!-- src/templates/Post.vue -->
<template>
  <Layout>
    <h1 v-html="$page.post.title" />
  </Layout>
</template>

<page-query>
query ($id: ID!) {
  post(id: $id) {
    title
  }
}
</page-query>
```

更多关于 Templates 的内容：https://gridsome.org/docs/templates/。

Layouts

布局是在页面和模板内部用于包装内容的Vue组件。布局通常包含页眉和页脚。

页面中通常按以下方式使用布局：

```
<template>
  <Layout>
    <h1>About us</h1>
  </Layout>
</template>

<script>
import Layout from '~/layouts/Default.vue'

export default {
  components: {
    Layout
  }
}
</script>
```

> 也可以在全球范围内使用布局，因此您无需每页导入它们。
>
> 请注意，Gridsome CLI创建的默认模板将使用全局布局组件。

更多关于 Layouts 的内容：https://gridsome.org/docs/layouts/。

Images

Gridsome具有内置的 `<g-image>` 组件，可输出优化的逐行图像。如果更改宽度和高度，则在开发时还可以实时调整大小和裁剪。 `<g-images>` 创建一个超小型模糊的嵌入式base64图像，然后在视图中使用IntersectionObserver延迟加载图像。

更多关于 Images 的内容：https://gridsome.org/docs/images/。

Linking

Gridsome具有内置的 `<g-link>` 组件，该组件在查看链接时使用 IntersectionObserver 来预取链接的页面。这使得在 Gridsome 站点中浏览非常快，因为单击的页面已经下载。

更多关于 `<g-link>` 的内容：https://gridsome.org/docs/linking/。

部署

> https://gridsome.org/docs/deployment/

三、Gridsome 基础

目录结构

```
.
├── package.json # 包说明文件
├── gridsome.config.js # Gridsome 配置文件
├── gridsome.server.js # 自定义 Gridsome 编译
├── static/ # 静态资源存储目录，该目录中的资源不做构建处理
└── src/
    ├── main.js # 应用入口
    ├── index.html # 公共页面
    ├── App.vue # 根组件
    ├── layouts/ # 布局组件
    │   └── Default.vue
    ├── pages/ # 路由页面
    │   ├── Index.vue
    │   └── Blog.vue
    └── templates/ # 模板
        └── BlogPost.vue
```

项目配置

Gridsome需要 gridsome.config.js 才能工作。插件和项目设置位于此处。基本配置文件如下所示：

```
module.exports = {
  siteName: 'Gridsome',
  siteUrl: 'https://www.gridsome.org',
  plugins: []
}
```

属性	类型	默认值	说明
siteName	string	`<dirname>`	该名称通常在标题标签中使用。
siteDescription	string	`''`	页面描述， `<meta name="description" content="xxx">`
pathPrefix	string	`''`	Gridsome假定您的项目是从域的根目录提供的。 如果您的项目将托管在名为my-app的子目录中， 则将此选项更改为“ / my-app”。
titleTemplate	string	`%s - <siteName>`	设置标题标签的模板。 %s占位符将替换为您在页面中设置的metaInfo的标题。
plugins	Array	`[]`	通过将插件添加到plugins数组来激活插件。
templates	object	`{}`	定义 collections 的路由和模板。
metadata	object	`{}`	将全局元数据添加到GraphQL模式。
icon	string | object	`'./src/favicon.png'`	Gridsome默认情况下会将位于src / favicon.png的任何图像用作favicon和touchicon， 但您可以定义其他路径或大小等。 图标应为正方形且至少16个像素。 网站图标将调整为16、32、96像素。默认情况下， 触摸图标的大小将调整为76、152、120、167、 180像素。
configureWebpack	object | Function		如果该选项是一个对象，它将与内部配置合并。
chainWebpack	Function		该函数将接收由webpack-chain驱动的ChainableConfig实例。

属性	类型	默认值	说明
runtimeCompiler	boolean	false	在运行时包括Vue模板编译器。
configureServer	Function		配置开发服务器。
permalinks.trailingSlash	boolean	true	默认情况下，在页面和模板后添加斜杠。启用此选项后，具有动态路由的页面将不包含尾部斜杠， 并且服务器上必须具有额外的重写规则才能正常工作。 另外，的静态路径不会自动包含尾部斜杠， 而应包含在路径中：
permalinks.slugify			使用自定义的Slugify方法。默认是 @sindresorhus/slugify
css.split	boolean	false	将CSS分成多个块。默认情况下禁用拆分。 拆分CSS可能会导致奇怪的行为。
css.loaderOptions	Object	{}	将选项传递给与CSS相关的 loader
host	string	localhost	
port	number	8080	
outputDir	string	`'dist'`	运行gridsome构建时将在其中生成生产构建文件的目录。

插件示例：

```
module.exports = {
  plugins: [
    {
      use: '@gridsome/source-filesystem',
      options: {
        path: 'blog/**/*.md',
        route: '/blog/:year/:month/:day/:slug',
        typeName: 'Post'
      }
    }
  ]
}
```

注意事项：

- 开发过程中修改配置需要重启服务

Pages 页面

页面负责在URL上显示您的数据。每个页面将静态生成，并具有自己的带有标记的index.html文件。

在Gridsome中创建页面有两种选择：

- 单文件组件
- 使用 Pages API 以编程方式创建页面

pages 中的单文件组件

`src/pages` 目录中的单文件组件将自动具有其自己的URL。文件路径用于生成 URL，以下是一些基本示例：

- `src/pages/Index.vue` becomes `/` *(The frontpage)*

- `src/pages/AboutUs.vue` becomes `/about-us/`
- `src/pages/about/Vision.vue` becomes `/about/vision/`
- `src/pages/blog/Index.vue` becomes `/blog/`

> 大小自动转小写，驼峰命名会自动使用短横杠分割

src/pages 中的页面通常用于诸如 /about/ 之类的固定 URL，或用于在 /blog/ 等处列出博客文章。

使用 Pages API 创建页面

可以使用 `gridsome.server.js` 中的 `createPages` 钩子以编程方式创建页面。如果您要从外部 API 手动创建页面而不使用 GraphQL 数据层，则此功能很有用。

```
module.exports = function (api) {
  api.createPages(({ createPage }) => {
    createPage({
      path: '/my-page',
      component: './src/templates/MyPage.vue'
    })
  })
}
```

动态路由

动态路由对于仅需要客户端路由的页面很有用。例如，根据URL中的细分从生产环境中的外部API获取信息的页面。

通过文件创建动态路由

动态页面用于客户端路由。可以通过将名称包装在方括号中来将路由参数放置在文件和目录名称中。例如：

- `src/pages/user/[id].vue` becomes `/user/:id`.
- `src/pages/user/[id]/settings.vue` becomes `/user/:id/settings`.

注意事项：

- 在构建时，这将生成 `user/_id.html` 和 `user/_id/settings.html`，并且您必须具有重写规则以使其正常运行。
- 具有动态路由的页面的优先级低于固定路由。例如，如果您有一个 `/user/create` 路由和 `/user/:id` 路由，则 `/user/create` 路由将具有优先级。

这是一个基本的页面组件，它使用路由中的id参数来获取客户端的用户信息：

```
<template>
  <div v-if="user">
    <h1>{{ user.name }}</h1>
  </div>
</template>

<script>
export default {
  data() {
    return {
      user: null
    }
  },
  async mounted() {
    const { id } = this.$route.params
    const response = await fetch(`https://api.example.com/user/${id}`)

    this.user = await response.json()
  }
}
</script>
```

始终使用 `mounted` 来获取客户端数据。由于在生成静态HTML时执行数据，因此在 `created` 中获取数据会引起问题。

通过编程方式创建动态路由

以编程方式创建带有动态路由的页面，以获取更高级的路径。动态参数使用 ： 来指定。

每个参数都可以具有一个自定义的正则表达式，以仅匹配数字或某些值。

```
module.exports = function (api) {
  api.createPages(({ createPage }) => {
    createPage({
      path: '/user/:id(\\d+)',
      component: './src/templates/User.vue'
    })
  })
}
```

生成重写规则

Gridsome无法为动态路由的每种可能的变体生成HTML文件，这意味着直接访问URL时最有可能显示404页。而是，Gridsome生成一个HTML文件，该文件可用于重写规则。例如，类似/ user /：id的路由将生成位于/user/_id.html的HTML文件。您可以具有重写规则，以将所有与/ user /：id匹配的路径映射到该文件。

由于每种服务器类型都有自己的语法，因此必须手动生成重写规则。 afterBuild 挂钩中的 redirects 数组包含应生成的所有必要的重写规则。

```
const fs = require('fs')

module.exports = {
  afterBuild ({ redirects }) {
    for (const rule of redirects) {
      // rule.from   - The dynamic path
      // rule.to     - The HTML file path
      // rule.status - 200 if rewrite rule
    }
  }
}
```

页面 meta 信息

Gridsome 使用 vue-meta 处理有关页面的元信息。

```
<template>
  <div>
    <h1>Hello, world!</h1>
  </div>
</template>

<script>
export default {
  metaInfo: {
    title: 'Hello, world!',
    meta: [
      { name: 'author', content: 'John Doe' }
    ]
  }
}
</script>
```

自定义 404 页面

创建一个 `src/pages/404.vue` 组件以具有一个自定义 404 页面。

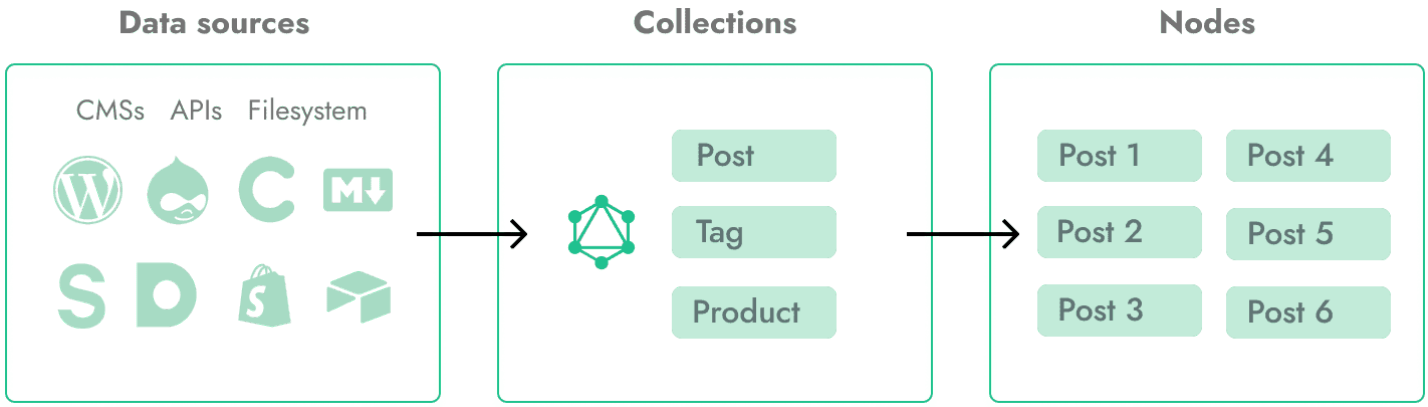
Collections 集合

集合是一组节点，每个节点都包含带有自定义数据的字段。如果您要在网站上放置博客文章，标签，产品等，则集合很有用。

添加集合

集合可以通过 source plugins 添加，也可以使用 Data Store API 自己添加。

在开发和构建期间，这些集合存储在本地内存数据存储中。节点可以来自本地文件（Markdown，JSON，YAML等）或任何外部API。



使用 source plugins 添加集合

将集合添加到 Gridsome 的最简单方法是使用源插件。本示例从 WordPress 网站创建集合。源插件的 typeName 选项通常用于为插件添加的集合名称添加前缀。

```
// gridsome.config.js
module.exports = {
  plugins: [
    {
      use: '@gridsome/source-wordpress',
      options: {
        baseUrl: 'YOUR_WEBSITE_URL',
        typeName: 'WordPress',
      }
    }
  ]
}
```

你可以在这里浏览插件列表。

使用 Data Store API 添加集合

您可以从任何外部 API 手动添加集合。

本示例创建一个名为 Post 的集合，该集合从 API 获取内容并将结果作为节点添加到该集合中。

```
// gridsome.server.js
const axios = require('axios')

module.exports = function (api) {
  api.loadSource(async actions => {
    const collection = actions.addCollection('Post')

    const { data } = await axios.get('https://api.example.com/posts')

    for (const item of data) {
      collection.addNode({
        id: item.id,
        title: item.title,
        content: item.content
      })
    }
  })
}
```

了解有关 Data Store API 的更多信息。

GraphQL 中的集合

每个集合将向 GraphQL schema 添加两个根字段，这些根字段用于检索页面中的节点。

字段名称是根据集合名称自动生成的。如果您将集合命名为 Post，那么在架构中将具有以下可用字段：

- `post` 通过 ID 获取单个节点。
- `allPost` 获取节点列表（可以排序和过滤等）。

自动生成 schema

探索可用的类型和字段

您可以通过在 GraphQL 资源管理器中打开架构选项卡来浏览可用字段。

阅读有关如何在 GraphQL 中查询节点的更多信息：https://gridsome.org/docs/querying-data/。

集合模板

模板用于为集合中的节点创建单个页面。节点需要相应的页面才能显示在其自己的URL上。

Templates

模板用于为集合中的节点创建单个页面。节点需要相应的页面才能显示在其自己的URL上。

设置模板

将数据添加到模板

节点字段作为 meta info

Layouts

Components

Linking

动态路由

图片处理

页面 Head 管理

添加全局头部元数据

将 meta data 添加到 pages 和 templates

从子组件覆盖 meta data

可用属性

Property	Description	Link
style	Adds a style tag	Docs
script	Adds a script tag	Docs
meta	Adds a meta tag	Docs
title	Changes title text	Docs
titleTemplate	Dynamic title text	Docs
link	Adds a link tag	Docs

环境变量

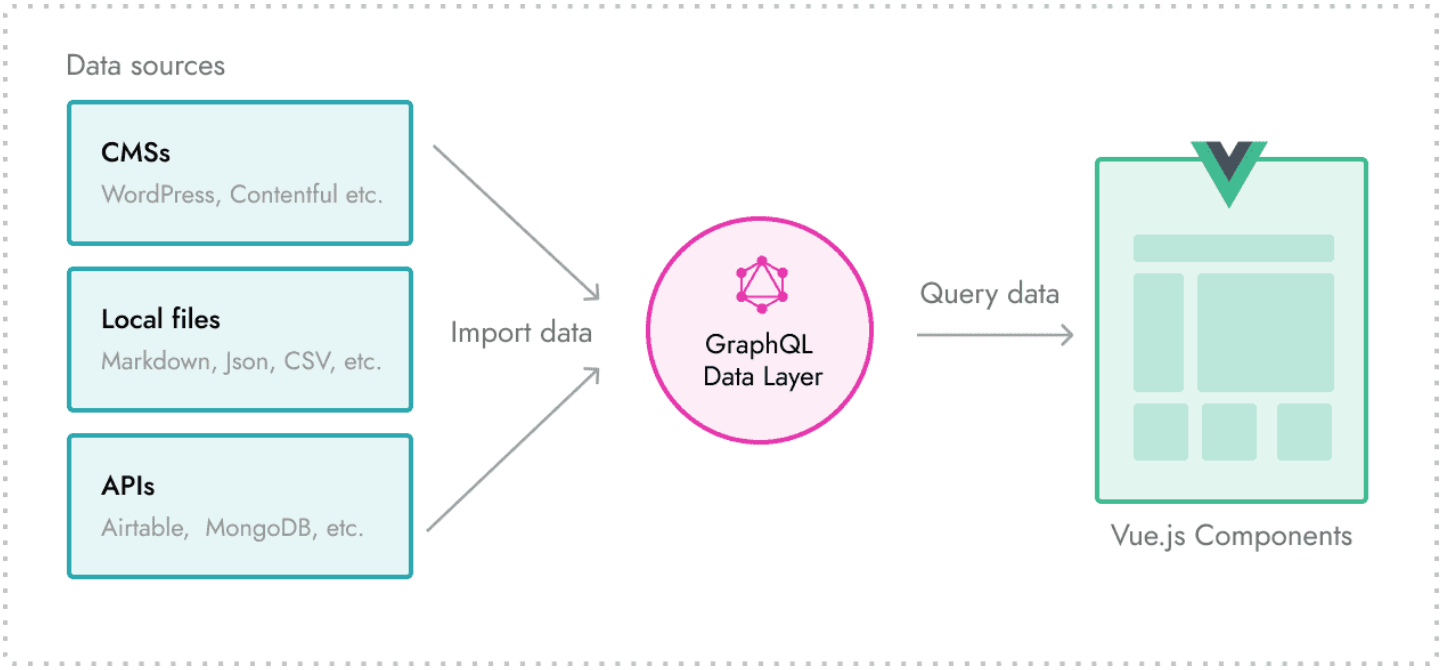
四、处理数据

上一篇介绍了选择 Gatsby 的原因，其中提到了 Gatsby 使用 GraphQL 。大家可能会有疑惑，不是建静态博客么，怎么会有 GraphQL？难道还要部署服务器？

其实这里 GraphQL 并不是作为服务器端部署，而是作为 Gridsome 在本地管理资源的一种方式。

通过 GraphQL 统一管理实际上非常方便，因为作为一个数据库查询语言，它有非常完备的查询语句，与 JSON 相似的描述结构，再结合 Relay 的 Connections 方式处理集合，管理资源不再需要自行引入其它项目，大大减轻了维护难度。

GraphQL数据层



GraphQL数据层是在开发模式下可用的工具。这是临时存储到 Gridsome 项目中的所有数据的地方。可以将其视为可帮助您更快更好地处理数据的本地数据库。

来自 GraphQL 数据层的数据将生成为静态内容。

数据层和导入数据的源之间没有实时连接。这意味着您需要重新生成网站以获取最新的数据更新。

如果需要动态数据，则应使用客户端数据。

> 提示：默认情况下，Pages 也 Site metadata 已添加到数据层。

处理数据

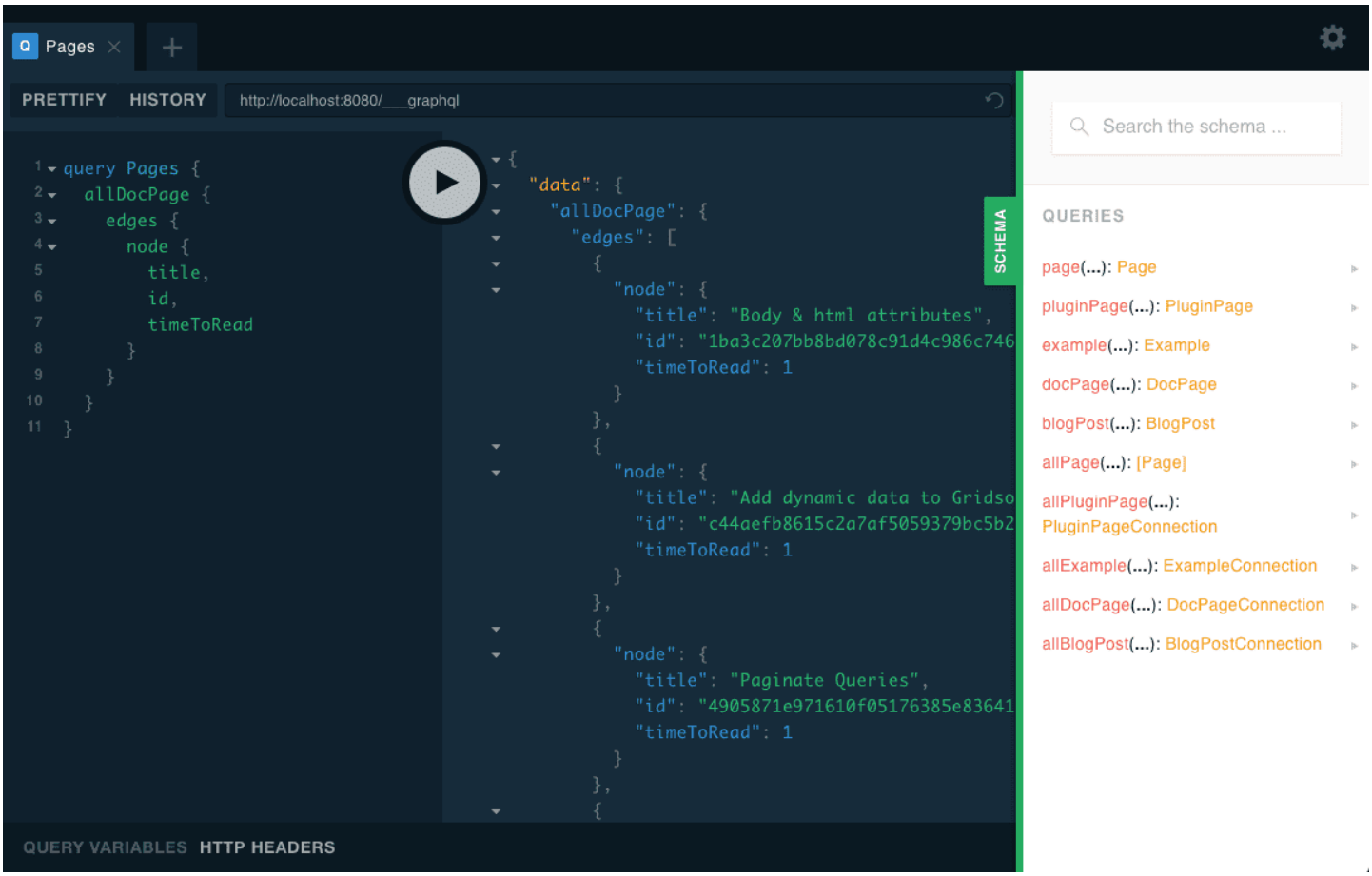
- How to import data.
- How to query data.
- How to filter data.
- How to create taxonomy pages.
- How to paginate data.
- How to add client-side / dynamic data.

GraphQL资源管理器

每个 Gridsome 项目都有一个 GraphQL 资源管理器，可以在开发模式下使用它来探索和测试查询。

在这里，您还将获得所有可用 GraphQL 集合的列表。

通常可以通过转到 `http:// localhost:8080/___explore` 来打开它。



导入数据

Gridsome 使您可以将数据从任何数据源导入 GraphQL 数据层。

使用 source plugins

使用外部 API

使用本地文件

Markdown

Images

YAML

CSV

JSON

查询数据

您可以将数据从GraphQL数据层查询到任何页面，模板或组件中。在Vue组件中，使用 `<page-query>` 或 `<static-query>` 块添加查询。

- 在 Pages 和 Templates 中使用 `<page-query>`
- 在 Components 中使用 `<static-query>`

如何使用 GraphQL 查询

在 Gridsome 中使用 GraphQL 很容易，并且您不需要了解 GraphQL。

这是一个如何在页面的 `page-query` 中使用GraphQL的示例：

```
<template>
  <div>
    <div v-for="edge in $page.posts.edges" :key="edge.node.id">
      <h2>{{ edge.node.title }}</h2>
    </div>
  </div>
</template>

<page-query>
query {
  posts: allWordPressPost {
    edges {
      node {
        id
        title
      }
    }
  }
}
</page-query>
```

使用 GraphQL，您仅查询所需的数据。这使得处理数据更加容易和整洁。

- 查询总是从 `query` 开始
- 然后是 `Posts` （可以是任何东西）
- 然后写一些内容例如 `posts: allWordPressPost` 。
- `allWordPressPost` 是您要查询的GraphQL集合的名称。
- `posts:` 部分是可选的别名。
- 使用 `posts` 作为别名时，您的数据将位于 `$page.posts` （如果使用 `<static-query>` ，则为 `$static.posts` ）。否则，它将在 `$page.allWordPressPost` 上可用。

学习更多关于 GraphQL 查询的内容：https://graphql.org/learn/queries/。

数据过滤

页面分类

页面分页

全局 metadata

客户端数据

五、样式和资源

在 Gridsome 中使用 CSS

使用自定义字体

使用外部脚本

使用 SVG 图标