2020-10-16 3-4 答疑

1、在3-4的作业中，有个接口返回字段的健名是动态的，这种情况GraphQL 怎么进行查询呢?

在 Gridsome 中有两种方式来处理这种需求。

方式一：让服务端处理接口

- 客户端渲染
- 在 Gridsome 中预取请求

2、在3-4的作业中-首页数据，是具有关联性的，即需要先请求一个接口A，从接口A中获取到相关参数，在通过B接口获取真正展示的数据，这种情况 gridsome-GraphQL 应该怎么做呢?

两种方案：

- 客户端渲染
- 数据请求

准备

3、SSR 中，如何把 INITIAL_STATE 抽离文件，而不是在把 INITIAL_STATE插入到页面的 script 标签；如果缓存在内存中，使用 pm2 来管理进程，页面的请求与获取 vuex store 数据的请求不是同一个进程处理的话，就获取不到对应的数据

- 在生产环境中，Vue SSR 会自己移除 script 中的 **INITIAL_STATE** 状态内容
- 如果想要手动的插入元素标签，也可以手动资源注入：https://ssr.vuejs.org/guide/build-config.html#client-config

4、使用 vue-cli3 SSR 如何配置热更新

- 手动配置（自己搭建 Vue SSR - 19小节）
- Vue CLI 工具默认已集成

建议把 Vue CLI 升级到最新版。

5、关于 strapi 配置 cors 跨域的问题希望可以扩展一下

- https://strapi.io/documentation/v3.x/concepts/middlewares.html

strapi 是一个无头 CMS，说白了就是一个通用的后台管理系统，适合开发人员使用。
strapi 默认提供了 CROS 跨域，而且默认是 * 允许任何用户都能访问接口资源。

6、希望能详细说一下 Nuxt.js 中的 exact 功能

- https://router.vuejs.org/zh/api/#exact

```
<nuxt-link to="/">首页</nuxt-link>

<nuxt-link to="/">首页</nuxt-link>
```

nuxt-link 组件的使用方式和 Vue Router 中的 router-link 几乎是一样的。
被匹配到路由会往导航组件上设置一个 CSS 类名：`router-link-active`
例如：

- /a/123
  - <router-link exact to="/a">
  - <router-link class="router-link-active" to="/a/123">
- /b
- /c

- /abc

7、希望老师聊一聊 nuxt 的性能和如何优化如：（cdn、component-cache、链接预读的优化）

推荐资源：

- https://tflin.com/2020/05/13/%E6%B5%85%E8%B0%88%20vue%20%E5%89%8D%E7%AB%AF%E5%90%8C%E6%9E%84%E6%A1%86%E6%9E%B6%20nuxt%20%E5%8F%8A%E5%85%B6%E6%80%A7%E8%83%BD%E4%BC%98%E5%8C%96/

主要性能问题：

- Nuxt 服务端渲染应用最大的性能问题在于 Node 服务端渲染性能，模板转换是 cpu 密集型的操作，node 又是单线程的，并发一高，cpu 就会飙到 100%
- 客户端的每次 request 都会到 node 服务器中，触发后端渲染。渲染服务器引入 renderer 和相应的 vue 应用，根据 route 找到相应的组件和数据，拉组件再拉数据（可能是异步的），加载组件生产 DOM，然后再使用 renderToString 吐给 response。

主要优化：

- 缓存
  - 页面缓存
  - 组件缓存
  - API 数据缓存
- 服务器集群分布式（提高机器的处理数量）
- 控制好首屏模块个数，对返回的结果进行精简，最小化，保证吐出到浏览器的内容足够小。这就是前面说的并不要对所有模块都做 ssr，需要首屏呈现的/需要爬虫爬的，我们直出，其他部分做 CSR 就行了

缓存

1、组件缓存

nuxt 的组件级别的缓存，使用的是 Component Cache module 模块，将 `@nuxtjs/component-cache` 从 npm 中添加到依赖中，在配置文件 nuxt.config.js 做出如下配置：

```
{
  modules: [
    '@nuxtjs/component-cache',
    [
      '@nuxtjs/component-cache',
      {
        max: 10000,
        maxAge: 1000 * 60 * 60
      }
    ]
```

```
  ]
}
```

在需要缓存的组件中使用 `serverCacheKey` 函数来标识

```
export default {
  name: 'ReplyItem',
  serverCacheKey: props => props.postId,
  props: {
    postId: String
  }
}
```

`serverCacheKey` 函数所返回的 key 必须有足够的信息去标识该组件，返回常量将导致组件始终被缓存，这对纯静态组件是有好处的。同时，缓存的组件必须要有唯一的 `name` 属性。
但值得注意的是，使用组件级别缓存的时候，不要缓存单一实例的组件。应该缓存的是那些重复使用的展示型组件，如 `v-for` 下的重复组件，在我所写的项目中，我使用组件级别的缓存也主要是这一类，如帖子列表、新闻列表、评论列表等。
如果 render 在渲染组件的过程中，命中缓存，则直接使用缓存结果，所以一些情况不能使用组件级别的缓存：

- 可能拥有依赖 global 数据的子组件。
- 具有在渲染 context 中产生副作用的子组件。

2、数据的缓存
在 node 服务器向后端请求数据的时间，也会影响到渲染的时间，所以数据层，最好也要有缓存。如果从后端 api 拉取数据的时间需要 3 秒，那这 3 秒会直接反应在首屏渲染时间上。对于数据层的缓存，应该对那些不涉及用户信息和实时性要求不高的接口进行缓存。对于数据层的缓存，使用的是 lru-cache 这个模块。
示例代码：

```
import axios from 'axios'
import LRU from 'lru-cache'
import md5 from 'md5'

const options = {
  // 最大缓个数
  max: 1000,
```

```
  // 缓存周期
  maxAge: 1000 * 60 * 5 // 5 分钟的缓存
}

const cache = new LRU(options)

// 需要缓存数据的接口
const CACHE_API = ['v1/api/xxxx', 'v1/api/xxxx', 'v1/api/xxxx',
'v1/api/xxxx']

export function get(url, params) {
  const key = md5(url + JSON.stringify(params))
  // 只在服务端进行缓存
  if (process.server && cache.has(key)) {
    // 命中缓存
    return Promise.resolve(cache.get(key))
  }

  return axios
    .get(url, { params })
    .then(res => {
      // 只在服务端进行缓存
      if (process.server && CACHE_API.includes(url)) {
        cache.set(key, res.data)
      }
      return Promise.resolve(res.data)
    })
    .catch(err => throw err)
}
```

3、页面的缓存

不是每一请求都需要触发后端渲染的，当页面不涉及用户数据，就可以对整个页面进行缓存。url 命中缓存的时候，直接将缓存吐给 response ，不再触发一系列的渲染活动。在 nuxt 中，使用页面级别的缓存，使用的是服务器渲染中间件 serverMiddleware 。在这一层的缓存中可以使用 redis 进行缓存，在 nginx 层的时候就可以直接调用 redis 吐数据，缓存过期后再重新出发 node 渲染并重新缓存。当然，也可以缓存在内存中。

示例代码：

根目录下新建一个 `serverMiddleware/pageCache.js`

```
const LRU = require('lru-cache')

const options = {
  max: 1000,
  maxAge: 1000 * 60 * 5
}

// 需要进行页面级别缓存的路由
const CACHE_URL = ['/xxx', '/xxx', '/xxx']

const cache = new LRU(options)

export default function (req, res, next) {
  const url = req._parsedOriginalUrl
  const pathname = !!url.pathname ? url.pathname : ''

  if (CACHE_URL.includes(pathname)) {
    const existsHtml = cache.get(pathname)
    if (existsHtml) {
      // 不要忘了设置 Content-Type 不然浏览器有时候可能不会渲染而是触发下载文件
      res.writeHead(200, { 'Content-Type': 'text/html; charset=utf-8' })
      return res.end(existsHtml.html, 'utf-8')
    } else {
      res.original_end = res.end
      res.end = function (data) {
        if (res.statusCode === 200) {
          cache.set(pathname, { html: data })
        }
        res.original_end(data, 'utf-8')
      }
    }
  }
  next()
}
```

页面缓存渲染中间件写好之后，在配置文件 nuxt.config.js 中使用

```
module.exports = {
  // ...
  serverMiddleware: ['~/serverMiddleware/pageCache.js']
  // ...
}
```

8、希望介绍下大网站的ssr方案，比如京东、淘宝

大型网站目前都是后端做页面静态化。比如淘宝、京东。
简单来说就是：
当后端操作数据变化之后，它会在服务端动态生成静态网页，它所使用的技术还是传统的服务端，比如模板引擎。然后将生成的静态页面部署到 CDN 服务，供客户访问。

9、视频里讲的文章列表好多状态都放在地址上，看起来有点不友好。示例网站是存在内存里，进入详情页再返回列表页，页面就回到初始状态。关于这种状态记录有没有什么比较优雅的方案?

- 试想这样一个场景，你在第3页进入了某个文章的详情，你看完了然后后退了一下。
- 状态放到地址上是目的是为了刷新不受影响
- 如果你放到内存上一刷新或者后退就没了

10、能不能讲讲码云上怎么搞自动部署

- Gitee Pages：https://gitee.com/help/articles/4136
  - 类似于 GitHub Pages
  - 免费版不支持自动更新，只能手动更新。
  - 必须付费 99/每年，而且只能对单仓库有效
  - 优点：香港节点，访问速度比 GitHub Pages 速度快
- Jenkins for Gitee：https://gitee.com/help/articles/4193

11、线上代码怎么捕捉页面异常，并提交后台？让你设计一个sdk你怎么做?

> 我之前面腾讯被问到了这两个，页面异常捕捉的我查了一些资料，但是还有些不太深入，然后关于缓存的 ，我是完全背下来的，但是我不懂后台怎么操作的，想请老师解答下

关于前端异常监控：

- https://zhuanlan.zhihu.com/p/31979395
- https://github.com/MriLiuJY/monitorFE
- https://github.com/pf12345/error-moniter
- https://github.com/a597873885/webfunny_monitor

- 什么是异常
  - 语法错误
  - 运行时错误
- 如何捕获异常
  - try-catch
  - window.onerrer
  - window.onunhandledrejection
- 异常捕获常见问题
  - Script error
  - Source Map
    - 因为项目发布以后上线运行的是压缩混淆之后的代码
    - 如果想要让报错之类的日志信息能够拿到源代码信息，则需要构建 Source Map 支持文件
- 如何上报异常
  - 构建错误信息，发布请求
  - Source Map

问题2：

- 使用 Node 搭建测试环境

12、在asyncData中请求文章列表数据，与在客户端mounted生命周期方法中请求同一个接口数据，文章的 favorited 属性不一致。

例如，在某个页面对该文章点赞之后，刷新页面，asyncData方法获取到的该文章favorited属性为false，但如果同时使用mounted生命周期方法去获取该文章的favorited属性却是true，不知道为什么会不一致，而且根据实际操作，应该是mounted获取的数据是正确的。route更新后，asyncData重新调用，数据重新获取到了，但页面组件却没有用新数据重新渲染?检查页面的data，发现仍然是旧数据 - 在pages/profile的index.vue页面组件中，使用asyncData方法获取数据，渲染组件。 - 当路由为/profile/:username时，渲染My Articles文章列表； - 当路由为/profile/:username/favorites时，渲染Favorited Articles文章列表 - 但是在点击＜nuxt-link>＜/nuxt-link>实际切换路由时，例如从/profile/:username切换到/profile/:username/favorites时，通过查看返回数据发现数据的确是返回了新的，但查看组件树发现组件的data却没有得到更新，导致视图没有更新。 - 但在多次切换（一般第二次切换）时，组件的data却又会更新为新数据，视图也随之更新。 为什么asyncData返回的新数据没有和组件的data融合，从

而更新data呢？– 如果不通过路由切换，而是使用query切换视图，如/profile/:username?favorites=favorites时，是可以顺利切换视图的，不会出现上述情况。

该问题确实是 Nuxt 应用的 Bug。我的解决方案是用的自定义路由和
或者下面是原因：

```
1 提醒一下，当使用路由参数时，例如从 /user/foo 导航到 /user/bar，原来的组件实例会被复用。
2 因为两个路由都渲染同个组件，比起销毁再创建，复用则显得更加高效。
3 不过，这也意味着组件的生命周期钩子不会再被调用。
```

13、gridsome 如果不使用vercel部署的话，使用github actions部署，那当strapi的数据发生了变化时，有什么方法可以触发github actions的自动部署。

目前我知道的 GitHub Actions 不支持通过外部请求触发。
还有一个思路可以尝试一下：看一下 GitHub 有没有对外提供调用 Actions 服务的 API。
GitHub API 文档：https://docs.github.com/en。