**Algumas dicas para usar no VueRouter**

26 Jul 2016

Se você utiliza o [Vue.js](https://vuejs.org/) como uma *Single Page Application* **SPA** já deve conhecer uma ferramenta indispensável para o funcionamento que é o Router. O Vue tem seu Router official chamado [VueRouter](https://router.vuejs.org/en/installation.html), sua utilização é extremamente simples, porem, ele é poderosíssimo em funcionalidades e customização.

Se você ainda não o conhece, confira esse artigo [Introdutório sobre o VueRouter](http://www.vuejs-brasil.com.br/vue-router/)

A seguir você verá alguns conceitos, formas de utilizar e configurações essenciais que agilizará o seu uso e que ficará mais fácil de solucionar problemas que possam a vir acontecer no desenvolvimento de sua SPA com o **VueRouter**.

**URL Amigável**

Quando falamos de URL Amigável, estamos nos referindo a ser amigável com os mecanismos de buscas e com o visitante no caso, por não conter caracteres (como o **hashbang** /#!/) que dificulte a compreensão do usuário.

<!-- Amigável -->

<http://seusite.com/blog/meu-post>

<!-- Não tão Amigável -->

<http://seusite.com/#!/blog/meu-post>

<http://seusite.com/#!/?post=1>

OBS: Como mencionado acima é amigável para mecanismos de busca, porem existe ainda pouquíssimo suporte das ferramentas em indexar SPAs, mas isso já está mudando, o Google já suporta esse tipo de aplicação e você pode saber mais através desse tutorial [Indexando páginas de sua SPA Vue.js no Google](http://www.vuejs-brasil.com.br/indexando-paginas-no-google-de-sua-spa-vue-js/)

No *VueRouter* é simples modificar e começar a utilizar esse recurso, que por padrão vem desativado devido a incompatibilidade dos browsers em suportar as novas features do modo history do **HTML5 pushState**.

Para habilitar, basta somente definir **history** como true

const router = new VueRouter({

history: true

})

É necessário configurar devidamente o seu servidor web para suporta HTML5 pushstate. Alguns servidores que hospedam sites estáticos já são configurados devidamente, mas se for fazer a configuração manualmente, segue uns *snippets* para Apache e Nginx.

**Apache**

<IfModule mod\_rewrite.c>

RewriteEngine On

RewriteBase /

RewriteRule ^index\.html$ - [L]

RewriteCond %{REQUEST\_FILENAME} !-f

RewriteCond %{REQUEST\_FILENAME} !-d

RewriteRule . /index.html [L]

</IfModule>

**Nginx**

rewrite ^(.+)$ /index.html last;

**Configuração Node.js + Express**

Sugestão do nosso amigo Erick Petrucelli, obrigado :) ```javascript const express = require(‘express’); const serveStatic = require(‘serve-static’); const path = require(‘path’);

app = express(); app.use(serveStatic(path.join(\_\_dirname, ‘dist’))); app.use(‘/\*’, serveStatic(path.join(\_\_dirname, ‘dist’)));

const port = process.env.PORT || 8080; app.listen(port);

## Suprimir erros

Se você não quiser receber erros detectados no interior dos ganchos (hooks) de transição, é possível configurar facilmente no momento que é instanciado o VueRouter.

> Importante utilizar em produção juntamente com a opção que o Vue disponibiliza para desativar o modo debug.

```javascript

const router = new VueRouter({

//...

suppressTransitionError: true

})

**Customizar classes em links ativos**

Para definir um link no nosso HTML utilizando o VueRouter, recomenda-se utilizar a diretiva v-link.

<a v-link="{ path: '/foo' }" title="#">Foo</a>

Essa diretiva recebe automaticamente nomes de classes quando a rota atual é a pertencente ao link, a classe padrão é a .v-link-active.

Podemos customizar de duas maneiras os nomes das classes de links ativos:

* A primeira é no ato da criação da instância do VueRouter
* const router = new VueRouter({
* //...
* linkActiveClass: '-active'
* })
* A segunda é no próprio objeto definido na diretiva v-link através da propriedade activeClass
* <a v-link="{ path: '/foo', activeClass: '-active' }" title="#">Foo</a>

E a partir da v0.7.8 ou superior, é possível aplicar classes ativas em outro elemento. A diretiva *v-link* procura necessariamente um elemento pai (parent) mais próximo e que tenha o *v-link-active* para assim poder aplicar a classe.

<ul>

<li v-link-active>

<a v-link="{ path: '/foo' }" title="#">Foo</a>

</li>

</ul>

**Ação antes de cada transição de rota (beforeEach)**

É muito simples definir ações **antes** de cada transição através do gancho global **beforeEach**.  
router.beforeEach(hook)

Ela é acionada a cada hook de transição e é possível chamá-lo múltiplas vezes. Muito utilizado para verificar rotas que necessitam de autenticação.

Nesse exemplo definiremos uma simples e pouco grosseira forma de posicionar o scroll no topo da página a cada rota acionada. Grosseiro, mas você pode utilizar uma boa lib ou o próprio animate do jQuery para fazer um efeito legal no scroll.

const router = new VueRouter()

router.beforeEach(function () {

window.scrollTo(0, 0)

})

**Ação após cada transição de rota (afterEach)**

Como vimos a global *beforeEach* que é possível executar uma ação *antes* de cada rota de transição, podemos utilizar o **afterEach** para executar uma ação **após** cada rota de transição efetuada com sucesso.

Nesse exemplo mostrarei uma forma recomendada de setar pageview para o [Google Analytics](https://analytics.google.com/analytics/web/), mas que funciona com qualquer outra ferramenta digital de marketing.

Esse Hook foi implementado de maneira global no Vue-router inicialmente por esse propósito, você pode conferir a issue [Implement global Router.prototype.afterEach hook](https://github.com/vuejs/vue-router/issues/88)

Esse gancho recebe como parâmetro a **transition** e podemos utilizá-lo para indicar a rota acessada para o Google Analytics.

router.afterEach((transition) => {

if (window.ga) window.ga('send', 'pageview', transition.to.path)

})

Neste caso:

* Para evitar erros, verificamos se existe disponível a propriedade ga no objeto window;
* E logo após comunicamos ao Analytics, setando a pageview e informando a URL (Nesse caso, o path);

**Componentes são reutilizáveis**

O VueRouter reutiliza os componentes que são construídos a cada rota, porem é importante saber que na reutilização é chamado somente o **data hook**, que é o gancho responsável em buscar os dados que depende de URL.

<http://seusite.com/search?q=curso>

Se consultada novamente com outra query na pesquisa, por exemplo:

<http://seusite.com/search?q=tutorial>

O componente responsável (por exemplo, componente search) irá ser reutilizado, mas para buscar novos resultados referente à pesquisa, temos que utilizar o data hook.

OBS: Sabendo disso, caso algum comportamento do componente necessite acompanhar a URL para funcionar corretamente, não o utilize em outros ganchos.

**Não reutilizando os componentes**

Também é possível informar no *objeto route* de seu componente, que não deseja reutilizá-lo e que a cada mudança de rota, reconstrua o componente passando por toda fase de validação e ativação.

// my-componente.vue

export default {

// ...

route: {

activate () ...

data () ...

canReuse: false

}

}

O canReuse pode ser definido como um valor Boolean, ou utilizado como uma function que retorne um valor boolean. Por padrão o **canReuse** é true.

**Gerar path**

Se você deseja gerar algum path utilizando as configurações de seu mapeamento de rotas, semelhante ao que a diretiva v-link faz, ficou exposto a partir da v0.7.12 o método stringfyPath().

// Mapeamento de rotas

routes = {

'/pesquisar': {

name: 'search'

},

//...

}

// Componente

export default {

//...

methods: {

getQueryDefault () {

return this.$route.stringifyPath({

name: 'search',

query: {

q: 'best'

}

})

}

}

}

Percebe-se que passamos um objeto aproveitando o uso de rotas nomeadas. O valor de retorno desse método *getQueryDefault* será /pesquisar/?q=best

Um dia poderá ser muito útil para gerar path que não seja a atual rota.

Por fim, não esqueça que a documentação é bastante detalhada e atualizada. Se você tiver alguma dica que não foi abordada aqui, compartilhe conosco.

Até a próxima!