

Публикация на тему

Проектирование динамических страниц с помощью фреймворка Nuxt

Цель дисциплины - научить разрабатывать динамические страницы с помощью фреймворка Nuxt (по сути, это не новый фреймворк - а следующая ступень развития Vue)

Анотация

-

-

Автор

[Михалькевич Александр Викторович](#)

Публикация

Наименование Проектирование динамических страниц с помощью фреймворка Nuxt

Автор А.В.Михалькевич

Специальность Цель дисциплины - научить разрабатывать динамические страницы с помощью фреймворка Nuxt (по сути, это не новый фреймворк - а следующая ступень развития Vue),

Анотация

-

Anotation in English

-

Ключевые слова nuxt, vue, pages, Проектирование динамических страниц, PDO, ПДО, фреймворк Nuxt.js, nuxt.js

Количество символов 8675

Содержание

[Введение](#)

1 [Начало проекта](#)

1.1 [Файл package.json](#)

1.2 [Зависимости проекта](#)

1.3 [Запуск проекта](#)

2 [Структура проекта](#)

2.1 [pages](#)

- 2.2 [components](#)
 - 2.3 [assets](#)
 - 2.4 [public](#)
 - 2.5 [plugins](#)
 - 2.6 [layouts](#)
 - 2.7 [nuxt.config.js](#)
 - 3 [Шаблонизация проекта и главная страница](#)
 - 3.1 [Маршрутный компонент главной страницы](#)
- [Заключение](#)
- [Список использованных источников](#)
- [Приложения](#)

Введение

1 Начало проекта

Прежде чем приступать к созданию приложения, необходимо убедиться в наличии следующих серверных технологий: node, npm, npx.

Используем консольные команды:

```
node -v
npm -v
npx -v
```

Далее можно переходить к созданию начальной структуры проекта (папок и файлов).

1.1 Файл package.json

Создаем папку для будущего проекта, в ней файл package.json со следующим содержимым:

```
{
  "name": "project name",
  "scripts": {
    "dev": "nuxt"
  }
}
```

После чего устанавливаем необходимые зависимости.

Список всех исполняемых скриптов и зависимостей приложения хранятся в этом файле.

1.2 Зависимости проекта

Можем начать с установки зависимости nuxt. Nuxt нужен для реализации маршрутов для динамических страниц.

Установим зависимость nuxt

```
npm i nuxt
```

После чего снова взглянем на файл `package.json`

```
{
  "name": "project name",
  "scripts": {
    "dev": "nuxt"
  },
  "dependencies": {
    "nuxt": "^3.2.0"
  }
}
```

Зависимости проекта хранятся в блоке `dependencies`.

1.3 Запуск проекта

```
npm run dev
```

Теперь главная страница сайта доступна по порту 3000

<http://localhost:3000>

2 Структура проекта

Продолжим разработку проекта, созданием следующих папок: `pages`, `components`, `assets`, `public`, `plugins`, `layouts`. Сделать это можно одной командой:

```
mkdir pages components assets public plugins layouts
```

Также в корне проекта создаём файл `nuxt.config.js`:

```
touch nuxt.config.js
```

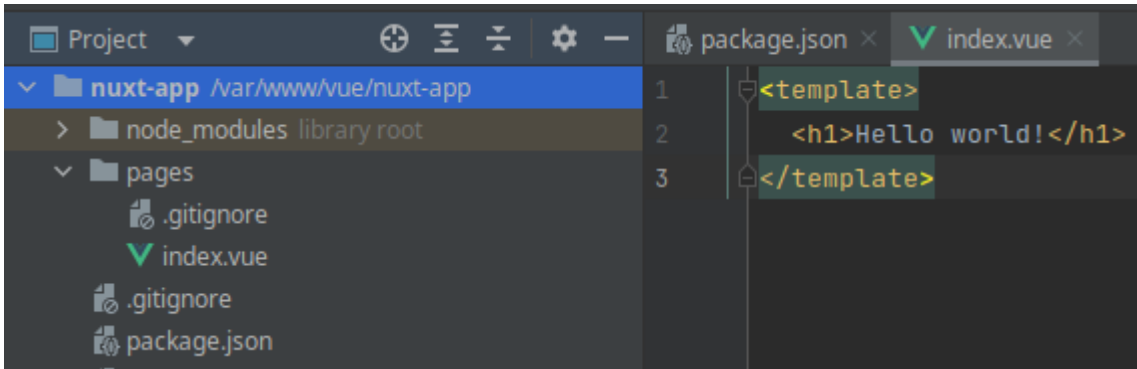
Рассмотрим начальную структуру проекта. Подробнее о каждой папке.

2.1 pages

Директория `pages` содержит маршрутные компоненты. Каждому файлу с расширением `.vue` этой папки Nuxt формирует соответствующий маршрут.

Маршруту главной страницы соответствует компонент `index.vue`. Все маршрутные компоненты создаём в этой папке. Например: `about.vue`, `contacts.vue` ...

Маршруты Nuxt формирует автоматически исходя из названий компонентов в папке `pages`. Поэтому создаём соответствующую папку, а в ней - компонент главной страницы - файл `index.vue`



Для создания другого маршрута - достаточно создать лишь его компонент в папке pages. Ссылку на такой маршрут лучше формировать с помощью специального в тэга NuxtLink.

2 .2 components

Дирректория components - папка для хранения маршрутных компонентов Vue.js. Здесь мы можем создавать свои компоненты и импортировать файлы .vue.

2 .3 assets

Данный каталог содержит некомпиллированные активы, такие как файлы Stylus или Sass, изображения или шрифты. В template ссылка на изображение может выглядеть так:

```
src="~/assets/your_image.png"
```

Подробнее - по ссылке [Документация Nuxt](#)

2 .4 public

Каталог public воспринимается клиентом как корневой. По сути, это папка для хранения любых файлов, которые надо хранить в открытом доступе, например иконку приложения и карту сайта.

В этой папке создадим два файла robots.txt и favicon.ico. Тогда

/static/robots.txt будет доступен по адресу <http://localhost:3000/robots.txt>

/static/favicon.ico будет доступен по адресу <http://localhost:3000/favicon.ico>

Если не собираетесь компилировать свои изображения и стили, то эта папка также подходит и для стилей и изображений. Тогда ссылка на изображение будет выглядеть так:

```
src="/my-image-2.png"
```

2 .5 plugins

Автоматическое подключение плагинов из этой папки в проект. Сперва необходимо создать плагин в этой папке. Затем подключить его в файле nuxt.config.js.

Для реализации запросов на бэкенд, можно воспользоваться Axios.

Установка модуля:

```
npm install @nuxtjs/axios
```

или

```
npm install axios
```

После чего, создайте файл `axios.js` со следующим содержимым:

```
import axios from 'axios';
//import {store} from '@store/index';
export default defineNuxtPlugin(nuxtApp => {
  //console.log(store.state.token);
  axios.defaults.baseURL = 'http://localhost:8000/api/';
  axios.defaults.headers["content-type"] = "application/json";
  axios.defaults.headers.common.authorization = `Bearer `;
  axios.defaults.headers.common['Access-Control-Allow-Origin'] = '*';
});
```

Подключение плагинов в файле `nuxt.config.js`

```
export default {
  css: ['~/assets/css/main.css'],
  modules: ['@nuxtjs/axios'],

  plugins: ['~/plugins/axios.js']
}
```

2 .6 layouts

Папка `layouts` предназначена для хранения компонента базового шаблона.

Подробнее по ссылке <https://nuxt.com/docs/guide/directory-structure/layouts>

2 .7 nuxt.config.js

В корне проекта должен находиться файл `nuxt.config.js` Его содержимое зависит от модулей и настроек проекта. Вот пример:

```
export default defineNuxtConfig({
  app: {
    head: {
      title: 'Учебный проект',
      meta: [
        { name: 'description', content: 'Здесь 2-3 предложения
связного текста о проекте' }
      ],
    }
  },
  css: ['~/assets/css/main.css'],
```

```
plugins: ['~/plugins/axios.js'],  
modules: ['@vuestic/nuxt', '@nuxt/ui'],  
})
```

3 Шаблонизация проекта и главная страница

Прежде всего необходимо создать главную страницу и заложить основу для создания других страниц приложения.

3.1 Маршрутный компонент главной страницы

В папке pages создадим компонент index.vue - это и будет компонент главной страницы. С содержимым главной страницы поместим в тэг template.

Заключение

Список использованных источников

Приложения