Nuxt实践

官方文档

Nuxt.js 是一个基于 Vue.js 的通用应用框架

通过对客户端/服务端基础架构的抽象组织，Nuxt.js 主要关注的是应用的UI渲染

Nuxt特性

1. 支持SSR
2. 依照文件路径自动生成路由
3. 静态文件服务
4. 代码分层

Nuxt安装

```
npm init nuxt-app <project-name>
```

路由

路由生成

pages目录中所有*.vue文件自动生成应用的路由配置

```
pages/
--| user/
-----| index.vue
-----| one.vue
--| index.vue
```

等价于

```
router: {
    routes: [{
            name: 'index',
            path: '/',
            component: 'pages/index.vue'
        },
        {
            name: 'user',
            path: '/user',
            component: 'pages/user/index.vue'
        },
        {
```

```
            name: 'user-one',
            path: '/user/one',
            component: 'pages/user/one.vue'
        }
    ]
}
```

动态路由

以下划线作为前缀的 .vue文件或目录会被定义为动态路由

```
pages/
--|detail/
----|_id.vue
```

等价于

```
{
    path: "/detail/:id?",
    component: _9c9d895e,
    name: "detail-id"
}
```

嵌套路由

创建内嵌子路由，你需要添加一个 .vue 文件，同时添加一个与该文件同名的目录用来存放子视图组件

```
pages /
-- | detail /
-- -- | _id.vue
-- | detail.vue
```

等价于

```
{
    path: '/detail',
    component: 'pages/detail.vue',
    children: [{
        path: ':id?',
        name: "detail-id"
    }]
}
```

此时可以从路由的$paramas.id中获取参数

路由导航

nuxt-link

使用nuxt-link进行导航，等价于router-link

```
<template>
    <div>
        <ul>
            <li v-for="goodingoods" :key="good.id">
                <nuxt-link :to="` /detail/${good.id}`">
                    <span>{{good.text}}</span>
                    <span>￥{{good.price}}</span>
                </nuxt-link>
            </li>
        </ul>
    </div>
</template>
```

别名

n-link

NLink

NuxtLink

禁用预加载

```
<n-link no-prefetch>pagenotpre-fetched</n-link>
```

nuxt-child

使用nuxt-child进行嵌套展示，等价于router-view

```
<template>
    <div>
        <nuxt-child></nuxt-child>
    </div>
</template>
```

路由配置

要扩展Nuxt.js创建的路由，可以通过router.extendRoutes选项配置

```
//nuxt.config.jsexportdefault
{
    router: {
        extendRoutes(routes, resolve) {
            routes.push({
                name: "foo",
                path: "/foo",
                component: resolve(__dirname, "pages/custom.vue")
            });
        }
    }
}
```

布局

默认布局

layouts/default.vue为项目的默认布局

自定义布局

首先在layouts文件夹下创建布局文件夹

```
<template>
    <div>
        <nuxt />
    </div>
</template>
```

在使用文件中进行使用

```
export default {
    layout: 'blank'
}
```

页面

页面就是vue组件，但是Nuxt.js添加了特殊配置

```
exportdefault {
    head() {
        return {
            // 网页标题
            title: "课程列表",
            //vue-meta利用hid确定要更新meta
            meta: [{
```

```
                name: "description",
                hid: "description",
                content: "setpagemeta"
            }],
            // 标题图标
            link: [{
                rel: "favicon",
                href: "favicon.ico"
            }],
        };
    },
};
```

异步数据获取

asyncData方法使得我们可以在设置组件数据之前异步获取或处理数据

@nuxt支持使用axios

安装

```
npm install @nuxtjs/axios -S
```

配置

```
// nuxt.config.js
{
    modules: ['@nuxtjs/axios', ],
    axios: {
        proxy: true
    },
    proxy: {
        "/api": "http://localhost:8080"
    },
}
```

使用

```
export default {
    async asyncData({
        $axios,
        params,
        error
    }) {
        if (params.id) {
            //asyncData中不能使用this获取组件实例
```

```
            //但是可以通过上下文获取相关数据
            const {
                data: goodInfo
            } = await $axios.$get("/api/detail", {
                params
            });
            if (goodInfo) {
                return {
                    goodInfo
                };
            }
            error({
                statusCode: 400,
                message: "商品详情查询失败"
            });
        } else {
            return {
                goodInfo: null
            };
        }
    }
};
```

中间件

中间件会在一个页面或一组页面渲染之前运行我们定义的函数，常用于权限控制、校验等任务

声明中间件

```
// middleware/auth.js
export default function({
    route,
    redirect,
    store
}) {
    // 上下文中通过store访问vuex中的全局状态
    // 通过vuex中令牌存在与否判断是否登录
    if (!store.state.user.token) {
        redirect("/login?redirect=" + route.path);
    }
}
```

使用中间件

局部使用

```
<!--admin.vue-->
<script>
```

```
    export default {
        middleware: ['auth']
    }
</script>
```

全局使用

```
// nuxt.config.js
router: {
    middleware: ['auth']
},
```

状态管理

应用根目录下如果存在 store 目录，Nuxt.js将启用vuex状态树

插件

Nuxt.js会在运行应用之前执行插件函数，需要引入或设置Vue插件、自定义模块和第三方模块时使用

声明插件

接口注入，利用inject插件机制将服务接口注入组件实例

```
// plugins/api-inject.js
export default ({
    $axios
}, inject) => {
    inject("login", user => {
        return $axios.$post("/api/login", user);
    });
};
```

添加请求拦截器附加token

```
// plugins/interceptor.js
export default function({
    $axios,
    store
}) {
    $axios.onRequest(config => {
        if (store.state.user.token) {
            config.headers.Authorization = "Bearer " +
store.state.user.token;
        }
        return config;
```

```
    });
}
```

注册插件

```
// nuxt.config.js
plugins: [
    "@/plugins/api-inject",
    "@/plugins/interceptor"
],
```

nuxtServerInit

通过在store的根模块中定义nuxtServerInit方法，Nuxt.js调用它的时候会将页面的上下文对象作为第2个参数传给它

nuxtServerInit只能写在store/index.js

nuxtServerInit仅在服务端执行

```
export const actions = {
    nuxtServerInit({
        commit
    }, {
        app
    }) {
        const token = app.$cookies.get("token");
        if (token) {
            console.log("nuxtServerInit: token:" + token);
            commit("user/init", token);
        }
    }
};
```

此处使用了cookies操作模块

```
npm i -S cookie-universal-nuxt
```

发布部署

服务端渲染应用部署

先进行编译构建，然后再启动Nuxt服务

```
npm run build
npm start
```

静态应用部署

Nuxt.js可依据路由配置将应用静态化，使得我们可以将应用部署至任何一个静态站点主机服务商

```
npm run generate
```