服务端渲染

服务器知识：koa、node.js

SSR原理：

将同一个组件渲染为服务器端的 HTML 字符串，将它们直接发送到浏览器，最后将这些静态标记"激活"为客户端上完全可交互的应用程序。

应用场景：

1. 应用需要更好的 SEO
2. 应用需要更快的内容到达时间 (首屏渲染时间)，特别是对于缓慢的网络情况或运行缓慢的设备。

ssr的局限：

1. 更大的服务器端负载
2. 较高的学习成本
3. 一些外部扩展库使用会受限

nuxt安装

```
npx create-nuxt-app <项目名>
```

启动

```
npm run dev
```

目录结构

assets：资源目录 `assets` 用于组织未编译的静态资源如 `LESS`、`SASS` 或 `JavaScript`。

components：组件目录 `components` 用于组织应用的 Vue.js 组件。Nuxt.js 不会扩展增强该目录下 Vue.js 组件，即这些组件不会像页面组件那样有 `asyncData` 方法的特性。

layouts：布局目录 `layouts` 用于组织应用的布局组件。

middleware：`middleware` 目录用于存放应用的中间件。

pages：页面目录 `pages` 用于组织应用的路由及视图。Nuxt.js 框架读取该目录下所有的 `.vue` 文件并自动生成对应的路由配置。

plugins：插件目录 `plugins` 用于组织那些需要在 `根vue.js应用` 实例化之前需要运行的 Javascript 插件。

static：静态文件目录 `static` 用于存放应用的静态文件，此类文件不会被 Nuxt.js 调用 Webpack 进行构建编译处理。 服务器启动的时候，该目录下的文件会映射至应用的根路径 `/` 下。

store：`store` 目录用于组织应用的 Vuex 状态树 文件。 Nuxt.js 框架集成了 Vuex 状态树 的相关功能配置，在 `store` 目录下创建一个 `index.js` 文件可激活这些配置。

nuxt.config.js：`nuxt.config.js` 文件用于组织Nuxt.js 应用的个性化配置，以便覆盖默认配置。

约定优于配置

路由：pages目录中所有 `*.vue` 文件生成应用的路由配置

导航：

```
<nuxt-link to="/">首页</nuxt-link>
```

嵌套：制造一个.vue文件和文件夹同名

```
pages/
--| main/
--| main.vue
```

动态路由：文件名或者文件夹名称要带_

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

页面

自定义布局：

1. 创建layouts/main.vue

```
<template>
  <div>
    布局内容
    <nuxt/>
  </div>
</template>
```

2. 页面中引入pages/main.vue

```
export default {
    layout: 'main',
};
```

异步数据：

asyncData

- 它在组件创建之前执行，里面不要用this访问组件实例
- 第一个参数是上下文
- 可以在服务端也可以客户端都执行
- asyncData会和data融合

典型用法：

```
async asyncData({$axios, error}) {
    // asyncData时间点早于组件创建，所以不能用this访问组件实例
    const result = await $axios.$get('/api/goods')
    if (result.ok) {
      return {goods: result.goods}
    }
    // 错误处理
    error({statusCode: 400, message:'查询数据失败'})
},
```

接口准备：

1. 创建server/api.js

```
const router = require("koa-router")({ prefix: "/api" });
const goods = [
  { id: 1, text: "Web全栈架构师", price: 1000 },
  { id: 2, text: "Python架构师", price: 1000 }
];
router.get("/goods", ctx => {
  ctx.body = {
    ok: 1,
    goods
  };
});
router.get("/detail", ctx => {
  ctx.body = {
    ok: 1,
    data: goods.find(good => good.id == ctx.query.id)
  };
});
```

```
router.post("/login", ctx => {
  const user = ctx.request.body;
  if (user.username === "jerry" && user.password === "123") {
    // 将token存入cookie
    const token = 'a mock token';
    ctx.cookies.set('token', token);
    ctx.body = { ok: 1, token };
  } else {
    ctx.body = { ok: 0 };
  }
});

module.exports = router;
```

2. 修改服务器配置，server/index.js

```
const bodyparser = require("koa-bodyparser");
const api = require("./api");

const app = new Koa();
// 设置cookie加密秘钥
app.keys = ["some secret", "another secret"];

// Import and Set Nuxt.js options
// ...

async function start() {
  // Instantiate nuxt.js
  // ...

  // 解析post数据并注册路由
  app.use(bodyparser());
  app.use(api.routes());

  //...
}

start();
```

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。

首页重定向，创建middleware/index-redirect.js

```
export default function({route, redirect}) {
    if (route.path === '/') {
        redirect('/main')
    }
}
```

注册中间件，nuxt.config.js

```
router: {
    middleware: ['index-redirect']
},
```

插件

Nuxt.js会在运行Vue应用之前执行JS插件，需要引入或设置Vue插件、自定义模块和第三方模块时特别有用。

创建plugins/api-inject.js

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

注册插件，nuxt.config.js

```
plugins: [
    "@/plugins/api-inject"
],
```

vuex

用户登录及登录状态保存，创建store/user.js

```
export const state = () => ({
  token: ''
});

export const mutations = {
  init(state, token) {
    state.token = token;
  }
```

```
};

export const getters = {
  isLogin(state) {
    return !!state.token;
  }
};

export const actions = {
  login({ commit }, u) {
    return this.$login(u).then(({ token }) => {
      if (token) {
        commit("init", token);
      }
      return !!data.token;
    });
  }
};
```

使用状态，创建中间件middleware/auth.js

```
export default function({ redirect, store }) {
  console.log("token:" + store.state.user.token);
  // 通过vuex中令牌存在与否判断是否登录
  if (!store.state.user.token) {
    redirect("/login");
  }
}
```

注册该中间件，admin.vue

```
export default {
    middleware: 'auth'
}
```

状态初始化，store/index.js

```
export const actions = {
  nuxtServerInit({ commit }, { req }) {
    const token = req.ctx.cookies.get("token");
    if (token) {
      console.log("初始化token");
      console.log(token);

      commit("user/init", token);
    }
  }
};
```