https://umijs.org/zh-CN/docs

（一）介绍

（二）Umi 如何工作？

（三）快速上手

（四）目录结构

（1）基础Umi项目目录结构

.

├── package.json（包）

├── .umirc.ts 配置内置功能和插件

├── .env 环境变量

├── dist 存放打包文件

├── mock（模） 存放mock文件

├── public（公） 该目录下所有文件会被复制到dist

└── src（源）

├── .umi 存放临时文件

├── layouts/index.tsx 约定式路由时的全局布局文件

├── pages 存放路由组件

├── index.less（首）

└── index.tsx（首）

└── app.ts 配置运行时

（五）配置

（1）配置文件

可以写在.umirc.ts，也可以写在config/config.ts，.umirc.ts优先级更高，更推荐

（2）TypeScript 提示

（3）本地临时配置

（3.1）.umirc.ts会和.umirc.local.ts做 deep merge

（3.2）config/config.ts会和config/config.local.ts做 deep merge

（3.3）.local.ts优先级最高，比UMI\_ENV的配置更高。umi dev时有效。umi build 时无效。应添加到 .gitignore

（4）多环境多份配置

可以通过环境变量UMI\_ENV区分环境指定配置。

（六）运行时配置

（1）运行时配置和配置的区别是他跑在浏览器端。

（2）配置项

（2.1）patchRoutes（径）修改路由，不需要返回

export function patchRoutes({ routes }) {

}

（2.2）render（渲）覆写render

export function render(oldRender) {

}

（2.3）onRouteChange（变）初始加载和路由切换时使用

export function onRouteChange({ routes, matchedRoutes, location, action }) {

}

（2.4）rootContainer（根）修改交给react-dom渲染的根组件

export function rootContainer(lastRootContainer, args) {

return

}

args包含：

routes，路由

plugin，运行时插件

history，history实例

（七）路由

（1）配置路由

（1.1）配置文件中通过routes（路）进行配置，包括：

（1. 1.1）path（径）

（1.1.2）component（组）

路径可以是绝对路径

路径也可以是相对路径，从src/pages开始找，若指向src目录，可以使用../也可以使用@/推荐后者

（1.1.3）exact（正）默认false

（1.1.4）routes配置子路由，通过 props.children 渲染子路由

（1.1.5）redirect重定向

（1.1.6）wrappers配置高阶组件，通过 props.children 渲染子路由

（1.1.7）title配置标题

（2）页面跳转（页面命令式跳转）

import { history } from 'umi';

history.push('/list');

history.push('/list?a=b');

history.push({

pathname: '/list',

query: {

a: 'b',

},

});

history.goBack();

（3）hash 路由

（4）Link 组件（页面声明式跳转）

（4.1）<Link to="/">点击跳转到/</Link>

（4.2）Link用于单页应用内部跳转，a标签用于外部地址跳转

（5）路由组件参数

路由组件可通过props获取以下属性

（5.1）match（匹），包含params（参）、path（径）、url（油） 和 isExact（是）

（5.2）location（位）

（5.3）history（历）

（5.4）route（路）

（6）传递参数给子路由

（八）约定式路由

（1）如果没有routes配置式路由。会进入约定式路由，分析src/pages目录获取路由

（2）动态路由

[name]命名的文件或文件夹为动态路由，分析成:name

（3）动态可选路由，暂不支持

（4）嵌套路由

目录下有\_layout.js时会生成嵌套路由，通过 props.children 渲染子路由

（5）全局 layout

src/layouts/index.js为全局路由，通过 props.children 渲染子路由

（6）不同的全局layout

不支持根据路由配置全局layout

支持根据路由渲染全局layout

（7）404 路由

src/pages/404.js为404页面

分析成routes最后一项

[

{ component: '@/pages/404' }

]

（8）扩展路由属性

可以在代码层通过导出静态属性的方式扩展路由

（九）插件

（十）页面跳转

（十一）HTML 模板

（1）修改默认模板

src/pages/document.ejs若存在，作为默认模板

（2）配置模板

模板可通过context（上）获取到umi提供的变量，包含：

route路由信息，需要打包出多个静态HTML时（即配置了exportStatic时）有效

config配置信息

（十二）Mock 数据

（1）约定式 Mock 文件

（2）编写 Mock（假） 文件

export default {

'GET /url/name': objectOrArrayValue, // GET 可忽略

'POST /url/name': (req, res) => {

},

}

（3）配置 Mock

（4）如何关闭 Mock？

（4.1）通过配置

export default {

mock: false,

};

（4.2）通过环境变量

MOCK=none umi dev

（5）引入 Mock.js

Mock.js是生成模拟数据的三方库

（十三）环境变量

（1）设置环境变量

（1.1）命令行中添加

若同时支持OS和Windows，可使用三方工具cross-env

yarn add cross-env --dev

cross-env PORT=8000 umi dev

（1.2）.env中添加

PORT=8000

（2）设置环境变量

APP\_ROOT，项目根目录,只能命令行中添加

ANALYZE，分析bundle构成

BABEL\_POLYFILL，根据targets配置为目标浏览器打补丁

COMPRESS，压缩css和js，build 时有效

FORK\_TS\_CHECKER，开启 TypeScript 类型检查

FRIENDLY\_ERROR（错）

HMR，代码热更新

HTML，输出 HTML（文），build 时有效

HOST（主），默认是 0.0.0.0

PORT（端），默认是 8000

PROGRESS，打包进度条

SOCKET\_SERVER，指定用于 HMR 的 socket（套） 服务器

UMI\_ENV，区分环境指定配置

WATCH，监听文件变更

WATCH\_IGNORED（忽），监听node\_modules下的文件变更

（十四）命令行工具

（1）umi build（建）

（2）umi dev（开）

（3）umi generate即umi g，内置的生成器，内置的类型page，用于生成最简页面

（4）umi plugin，查看当前项目使用的所有umi插件

（5）umi help，查看帮助文档

（6）umi version即umi v，查看版本号

（7）umi webpack，查看webpack（歪）配置：

可选参数 说明

--rules 查看webpack.module.rules配置详情

--rule=[name] 查看webpack.module.rules中某个规则的配置详情

--plugins 查看webpack.plugins配置详情

--plugin=[name] 查看webpack.plugins中某个插件的配置详情

（十五）使用 CSS

（1）全局样式

src/global.css 为全局样式

（2）CSS Modules

自动识别 CSS Modules

（2.1）CSS Modules

import（导） cssFile from './cssFile.css';

（2.2）非 CSS Modules

import './cssFile.css';

（3）CSS 预处理器

支持css，less，不支持sass和stylus。若需要，可通过chainWebpack配置或插件支持。

（十六）使用图片

（1）JS 里使用图片

（1.1）<img src={require('相对路径')} />

（1.2）<img src={require('@指向src的相对路径')} />

（1.3）<img src={require('cdn绝对路径')} />

（2）CSS 里使用图片

（2.1）background: url(相对路径);

（2.2）background: url(~@指向src的相对路径);

（2.3）background: url(cdn绝对路径);

（3）Base64（背） 编译

存放在项目中的图片被引入时，若图片小于10K，编译为Base64，否则为图片文件。

10K可以通过inlineLimit配置修改。

（十七）按需加载

（1）启用按需加载

默认关闭，配置开启：

export default {

dynamicImport: {},

}

（2）使用按需加载

（2.1）按需加载组件

import { dynamic } from 'umi';

const 组件名 = dynamic({

loader: async function() {

return 组件;

},

});

（2.2）按需加载非组件

import('非组件').then(() => {

});

（十八）部署

（1）默认方案

构建后只输出index.html、umi.js 和 umi.css 三个文件

（2）不输出html文件

配置环境变量 HTML=none

（3）部署 html 到非根目录

部署在 /a/b/c下，配置：

export default {

base: '/a/b/c',

};

（4）用 hash（哈） history

export default {

history: { type: 'hash' },

};

（5）按需加载

（6）静态资源在非根目录或 cdn

（7）使用 runtime 的 publicPath（公）

在html里管理publicPath：

export default {

runtimePublicPath: true,

};

<script>

window.publicPath = <%= YOUR PUBLIC\_PATH %>

</script>

（8）静态化

每个路由都在对应目录下输出index.html，暂不支持动态路由

export default {

exportStatic: {},

}

（9）HTML 后缀

每个路由都在dist目录下输出index.html，暂不支持动态路由

export default {

exportStatic: {

htmlSuffix: true,

},

}

（10）静态化后输出到任意路径

export default {

exportStatic: {

htmlSuffix: true,

dynamicRoot: true,

},

}

目包配环打模公源临布路首首运

可本优多 运径渲变根

路径组正子重高标 命声参匹参径油是位历路

约动可嵌全不4扩 模上假关三

环根分补压检错热文主端进套环监忽

命建开生插帮版歪 全自导预图背需组非

输不非哈公每后任

目包配环打模公源临布路首首运

可本优多运径渲变根

路径组正子重高标命声参匹参径油是位历路

约动可嵌全不4扩模上假关三

环根分补压检错热文主端进套环监忽

命建开生插帮版歪全自导预图背需组非

输不非哈公每后任

目可路约环命输