目录及约定

在文件和目录的组织上，umi 更倾向于选择约定的方式。

一个复杂应用的目录结构如下：

```
.
├── dist/                          // 默认的 build 输出目录
├── mock/                          // mock 文件所在目录，基于 express
├── config/
│   ├── config.js                  // umi 配置，同 .umirc.js，二选一
└── src/                           // 源码目录，可选
    ├── layouts/index.js           // 全局布局
    ├── pages/                     // 页面目录，里面的文件即路由
    │   ├── .umi/                  // dev 临时目录，需添加到 .gitignore
    │   ├── .umi-production/       // build 临时目录，会自动删除
    │   ├── document.ejs           // HTML 模板
    │   ├── 404.js                 // 404 页面
    │   ├── page1.js               // 页面 1，任意命名，导出 react 组件
    │   ├── page1.test.js          // 用例文件，umi test 会匹配所有 .test.js 和 .e2e.j
    │   └── page2.js               // 页面 2，任意命名
    ├── global.css                 // 约定的全局样式文件，自动引入，也可以用 global.less
    ├── global.js                  // 可以在这里加入 polyfill
    ├── app.js                     // 运行时配置文件
├── .umirc.js                      // umi 配置，同 config/config.js，二选一
├── .env                           // 环境变量
└── package.json
```

ES6 语法

配置文件、mock 文件等都有通过 `@babel/register` 注册实时编译，所以可以和 src 里的文件一样，使用 ES6 的语法和 es modules 。

dist

默认输出路径，可通过配置 outputPath 修改。

mock

此目录下所有的 `.js` 文件（包括 `_` 前缀的）都会被解析为 mock 文件。

比如，新建 `mock/users.js` ，内容如下：

```js
export default {
  '/api/users': ['a', 'b'],
};
```

然后在浏览器里访问 http://localhost:8000/api/users 就可以看到 `['a', 'b']` 了。

如果想忽略 mock 文件夹下的部分文件，参考 mock.exclude 配置。

src

约定 `src` 为源码目录，如果不存在 `src` 目录，则当前目录会被作为源码目录。

比如：下面两种目录结构的效果是一致的。

```
+ src
  + pages
    - index.js
  + layouts
    - index.js
- .umirc.js
```

```
+ pages
  - index.js
+ layouts
  - index.js
- .umirc.js
```

src/layouts/index.js

全局布局，在路由外面套的一层路由。

比如，你的路由是：

```
[
  { path: '/', component: './pages/index' },
  { path: '/users', component: './pages/users' },
]
```

如果有 `layouts/index.js` ，那么路由就会变为：

```
[
  { path: '/', component: './layouts/index', routes: [
    { path: '/', component: './pages/index' },
    { path: '/users', component: './pages/users' },
  ] }
]
```

src/pages

> 注：配置式路由下无效。

约定 pages 下所有的 `js` 、 `jsx` 、 `ts` 和 `tsx` 文件即路由。关于更多关于约定式路由的介绍，请前往路由章节。

src/pages/404.js

404 页面。注意开发模式下有内置 umi 提供的 404 提示页面，所以只有显式访问 `/404` 才能访问到这个页面。

src/pages/document.ejs

有这个文件时，会覆盖默认的 HTML 模板。

模板里需至少包含根节点的 HTML 信息，

src/pages/.umi

这是 umi dev 时生产的临时目录，默认包含 `umi.js` 和 `router.js`，有些插件也会在这里生成一些其他临时文件。可以在这里做一些验证，**但请不要直接在这里修改代码，umi 重启或者 pages 下的文件修改都会重新生成这个文件夹下的文件。**

src/pages/.umi-production

同 `src/pages/.umi`，但是是在 `umi build` 时生成的，`umi build` 执行完自动删除。

.test.(js|ts) 和 .e2e.(js|ts)

测试文件，`umi test` 会查找所有的 `.test.js` 和 `.e2e.js` 文件来跑测试。

src/global.(js|ts)

此文件会在入口文件的最前面被自动引入，可以在这里加载补丁，做一些初始化的操作等。

src/global.(css|less|sass|scss)

此文件不走 css modules，且会自动被引入，可以在这里写全局样式，以及做样式覆盖。

src/app.(js|ts)

运行时配置文件，可以在这里扩展运行时的能力，比如修改路由、修改 render 方法等。

.umirc.(js|ts) 和 config/config.(js|ts)

编译时配置文件，二选一，不可共存。

.env

```
CLEAR_CONSOLE=none
BROWSER=none
```

> 这里定义的系统环境变量在整个 `umi-build-dev` 的生命周期里都可以被使用

.env.local

本地化的系统环境变量，该文件通常不用提交到代码仓库。本地启动时， 相同内容 `.env.local` 会覆盖 `.env` 。

在 GitHub 上编辑此页

Last Updated: 2019/6/26 下午3:45:35

← 例子 路由 →