guluS-脚手架

背景

通过脚手架快速搭建插件目录结构。

方案

通过 plop 脚手架工具，生成插件目录及必要文件，目录结构如下

```
├── README.md                    // 插件说明
├── __tests__                    // 单测
│   └── index.test.ts
├── app
│   └── extension                // gulu 扩展
│       ├── app.ts               // 添加 app.xxx
│       ├── context.ts           // 添加 ctx.xxx
│       ├── request.ts           // 添加 ctx.request.xxx
│       └── response.ts          // 添加 ctx.response.xxx
├── app.ts                       // 提供 beforeStart、afterStart 钩子
├── config                       // 配置文件
│   └── config.default.ts
├── index.d.ts                   // 类型声明
├── jest.config.js               // jest 配置文件
├── package.json                 // 默认 npm 包命名 @gulu-s/plugin-{{name}}
├── package.sh                   // 打包脚本
├── replacer.config.js           // 打包更改 json 工具
└── tsconfig.json
```

开发流程

安装依赖

> ❗ 后续插件开发需要注意，尽可能将依赖包抽离到根目录管理。

首次拉取仓库 - guluS-plugins，在根目录下安装依赖。

```
yarn setup
```

创建新插件

在根目录下，通过 plop 命令搭建插件目录结构，会在 `/plugins/xxx` 生成插件相关目录文件

```
yarn create-plugin
```

具体例子

```
yarn create-plugin
yarn run v1.22.18
$ plop
? 请输入插件文件夹名称（全小写，单词间以“-”连接） rpc-forward
? 请输入插件描述（可选）
? 请输入组件版本号 0.0.1
? 请输入开发者姓名 hooyah
✔  ++ /plugins/rpc-forward/package.json
✔  ++ /plugins/rpc-forward/package.sh
✔  ++ /plugins/rpc-forward/replacer.config.js
✔  ++ /plugins/rpc-forward/tsconfig.json
✔  ++ /plugins/rpc-forward/app.ts
✔  ++ /plugins/rpc-forward/index.d.ts
✔  ++ /plugins/rpc-forward/config/config.default.ts
✔  ++ /plugins/rpc-forward/app/extension/app.ts
✔  ++ /plugins/rpc-forward/app/extension/context.ts
✔  ++ /plugins/rpc-forward/app/extension/request.ts
✔  ++ /plugins/rpc-forward/app/extension/response.ts
✔  ++ /plugins/rpc-forward/README.md
✔  ++ /plugins/rpc-forward/__tests__/index.test.ts
✨  Done in 20.12s.
```

开发辅助工具

- 单测：接入 jest

> 📌 约定所有的单测文件都放置在 __tests__ 文件夹下

在插件目录下，执行单测命令

```
yarn test
```

大概会有如下输出

```
yarn test
yarn run v1.22.18
$ ../../node_modules/.bin/jest
 PASS  __tests__/index.test.ts
  guluS-plugin-rpc-forward
    ✓ mock xxx (1 ms)

----------|---------|----------|---------|---------|-------------------
File      | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
----------|---------|----------|---------|---------|-------------------
All files |       0 |        0 |       0 |       0 |
----------|---------|----------|---------|---------|-------------------
Test Suites: 1 passed, 1 total
Tests:       1 passed, 1 total
Snapshots:   0 total
```

```
Time:        3.585 s
Ran all test suites.
Force exiting Jest: Have you considered using `--detectOpenHandles` to detect async operations that kept running after all tests finished?
✨  Done in 5.20s.
```

- Eslint & Prettier

目前使用 gulu 那套 - @gulu/style

打包发布

在插件目录下，通过 package.sh 脚本进行打包及发布。

```
yarn package
```

package 脚本主要做了几件事：

1. 清除旧打包文件，通过 `tsc` 进行编译，输出结果到 `package` 目录
2. 复制必要的文件到 `package` 目录，如 package.json、index.d.ts
3. 通过 `replacer` 工具修改 package.json 版本号等。
4. 完成打包，进入 `package` 目录进行发布。npm 包命名默认为 `@gulu-s/plugin-{{name}}`

```
#!/bin/bash
set -e

echo "node version is $(node -v)"

BUILD_DIR="package"

rm -rf ${BUILD_DIR}

yarn build

# copy files
ls | grep -v ${BUILD_DIR} | xargs -I files rsync -a \
  --ignore-existing \
  --include="*.d.ts" \
  --include="package.json" \
  --include="*.md" \
  --exclude="node_modules" \
  --exclude="__tests__" \
  --exclude="scripts" \
  --exclude="coverage" \
  --exclude="**/*.ts" \
  --exclude="*.config.js" \
  --exclude="*.json" \
  --exclude="*.sh" \
  files ${BUILD_DIR}

yarn replacer

cd package
```

```
31
32  yarn publish
```

其他

bnpm 用户手册 | bnpm Docs

gulu 插件开发介绍

gulu 单测介绍