# 第三节 Plop



## 1.定义

- 一个小而没得脚手架工具
- 可以自动生成代码文件的工具。
- 根据命令行输入的信息和模板文件,自动生成指定的文件

## 2. 使用plop

- 将plop模块作为项目开发依赖安装
- 在项目根目录下创建一个plopfile.js文件
- 在plopfiles.js文件中定义脚手架任务
- 编写用于生产特定类型文件的模板
- 通过plop提供的cli运行脚手架任务

### 2.1 安装

```
// 全局安装
npm i -g plop

// 本地安装
npm i --save-dev plop
```

## 2.2配置文件

- Plop 入口文件为plopfile.js,需要导出一个函数
- 此函数接收一个plop对象,用于创建生成器任务

```
module.exports = plop => {
    plop.setGenerator("component", {
        description: 'create a component',  // 生成器的描述
        prompts: [                          // 发出的命令行问题
            {
                type: 'input',              // 问题的输入方式
                name: 'name',               // 问题返回值接收的key
                message: 'component name',  // 问题提示
                default: 'MyComponent'      // 问题的默认答案

            }
        ],
        actions:  [                         // 执行完命令后要执行的动作
            {
                type: 'add',                // 代表添加文件
                path: 'src/components/{{name}}/{{name}}.js',     // 添加文件的路径,使用双花括号插入值
                templateFile: 'plop-template/component.js.hbs' // 指定模板文件,使用hbs文件
            },
            {
                type: 'add',
                path: 'src/components/{{name}}/{{name}}.css',
                templateFile: 'plop-template/component.css.hbs'
            },
            {
                type: 'add',
                path: 'src/components/{{name}}/{{name}}.test.js',
                templateFile: 'plop-template/component.test.js.hbs'
            },
        ]
    })
}
```

## 2.3 执行命令

```
npx plop 生成器名称
```

## 2.4 plop参数名称

```
// 执行指定配置
plop 配置名称

// 执行指定配置并设置参数
plop 配置名称 输入参数

// 执行 plopfile 文件
--plopfile 文件路径

// 设置工作路径
--cwd

// 帮助
-h, --help

// 全局初始化
-i, --init

// 显示版本
-v, --version
```

## 2.5 generator 参数

生成器， 用来生成执行文件模板或向文件中加入模板信息

- description 描述生成器行为
- prompts 提示配置 详情
  - type 交互类型 `input` `number` `checkbox` ...
  - name 参数使用存储的属性名
  - message 提示信息
  - default 参数默认值
- actions 执行配置 详情

- type 预设类型 `add` `modify` `addMany` `etc`
- force
- data 返回给模板的数据
- abortOnFail 当有action 执行失败时， 是否终止其他 action

## 2.6 actions

- **addA** 创建文件
  - path 文件生成目录
  - template 模板字符串, 使用字符串模板生成文件内容

```
{
    template: '<h1>{{ title }} <h1>'
}
```

- templateFile 模板文件地址, 使用模板文件生成文件
- skipIfExists 如果文件已存在，将跳过
- force
- data 模板参数
- abortOnFail 当有action 执行失败时， 是否终止其他 action
- **addMany** 创建多个文件
  - destination
  - base 替换的基础目录

```
{
    destination: 'target',
    base: 'root/sub',
    templateFiles: 'root/sub/*.hbs'
}
// 生成的文件目录:  target/file.hbs
```

- templateFiles 模板文件匹配规则 参考

```
{
    templateFiles: 'plop-templates/view/*.hbs'
}
```

- globOptions 更改匹配方式
- stripExtensions
- verbose 是否打印所有文件目录
- skipIfExists

- force
- data
- abortOnFail
- **modify** 修改
  - path
  - pattern 替换规则 正则
  - template
  - templateFile
  - data
  - abortOnFail
- **append** 添加
  - path
  - pattern 插入规则 正则
  - unique
  - separator
  - template
  - templateFile
  - data
  - abortOnFail

## 2.7 模块分组

我们可将多个 配置分配到多个文件中单独管理

```
//  module/view/prompt.js 页面模板
const conf = {
    description: "view template",
    prompts: [
        {
            type: 'input',
            name: 'name',
            message: 'file name',
        }
    ],
    actions: data => {

        const name = '{{name}}'
        return [
            {
                type: 'add',
```

```
                path: `template/${name}.vue`,
                templateFile: 'plop-templates/view/index.hbs',
            }
        ]

    }
}

module.exports = function (plop){
    plop.setGenerator('view', conf)
}

//  module/components/prompt.js 组件模板
const conf = {
    description: "cmp template",
    prompts: [
        {
            type: 'input',
            name: 'name',
            message: 'file name',
        }
    ],
    actions: data => {

        const name = '{{name}}'
        return [
            {
                type: 'add',
                path: `template/${name}.vue`,
                templateFile: 'plop-templates/cmp/index.hbs',
            }
        ]

    }
}

module.exports = function (plop){
    plop.setGenerator('view', conf)
}

```

```
// root/plopfile.js
const viewCallback = require('./plop-templates/view/prompt')
const cmpCallback = require('./plop-templates/cmp/prompt')

module.exports = function(plop){
    cmpCallback(plop)
    viewCallback(plop)
}
```

## 2.8 Handlebars 语法规则

Handlebars 模板语法