# 溯源脚手架实现方案

1. 实现目的

1.开发过程中的业务类型多，一套代码到处复制，不利于维护，最好是通过场景选择，动态匹配业务需求

2.代码规范没有办法统一，难维护

基于上面的理由，建立一个可以根据业务需求动态配皮输出的工具非常有必要

1. 具体实现

1.需要用到的模块，已经模块介绍

****commander**** ：参数解析 --help其实就借助了他~

****inquirer**** ：交互式命令行工具，有他就可以实现命令行的选择功能

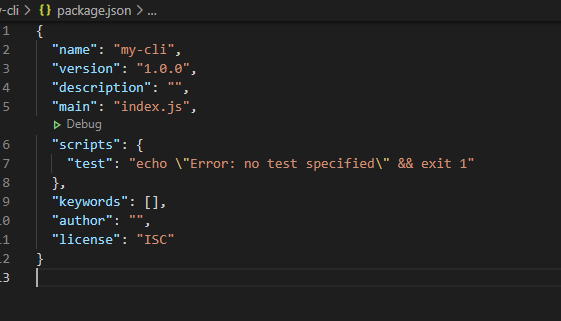
****download-git-repo**** ：在git中下载模板

****chalk**** ：粉笔帮我们在控制台中画出各种各样的颜色

****metalsmith**** ：读取所有文件，实现模板渲染

****consolidate**** ：统一模板引擎

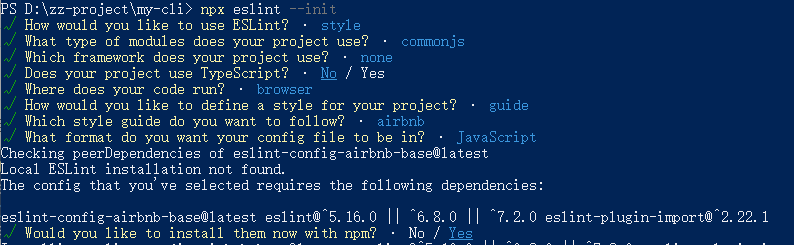
1. 创建工程
2. 新建目录my-cli，cd my-cli 输入命令npm init -y，新建packjson.json文件



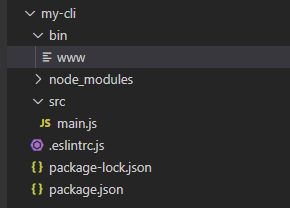
1. 开启exlint,规范代码，

cnpm i eslint

npx eslint --init

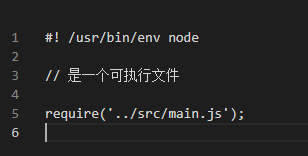


1. 在my-cli下面新建bin,src文件夹，bin下面新建文件www，src文件夹下新建main.js



在www上面，添加一行#! /usr/bin/env node，表示这个文件用node运行，引入main.js

www上面最终代码：



main.js上面添加console.log(‘test’)测试输出

1. 建立npm link 链接，链接包到全局下使用，在packjson.json上面，配置bin



在my-cli目录下面输入命令npm link,建立全局链接，输入命令suyuan-cli,测试输出：test

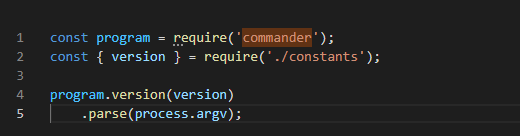
解析，这个npm link 会在全局的范围里注册命令suyuan-cli,当执行到这个命令的时候就会执行后面对应的文件，在发布到npm，拉去代码之后，npm同样会看这里有没有配置bin字段，有的话就全局注册，所以在全局cmd的任何位置，只要输入命令就可以执行www文件了。

1. 解析命令行参数

我们要解决当前用户输入的参数，例如输入suyuan-cli create my-project的时候要知道后面的两个参数，这个时候就要用到这个commander模块了

cnpm install commander -D,安装模块依赖

在main.js上面配置下面代码：



在src下面新增文件夹constants.js,代码配置如下，引入版本号



这个时候在目录输入命令suyuan-cli -V



显示出1.0.0证明配置模块成功

1. 配置指令命令

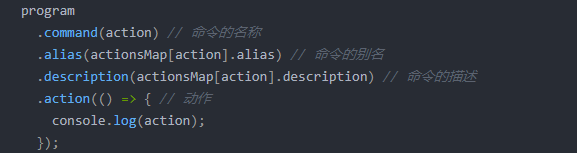
program有以下几个函数用于配置命令：

\*command//命令名称

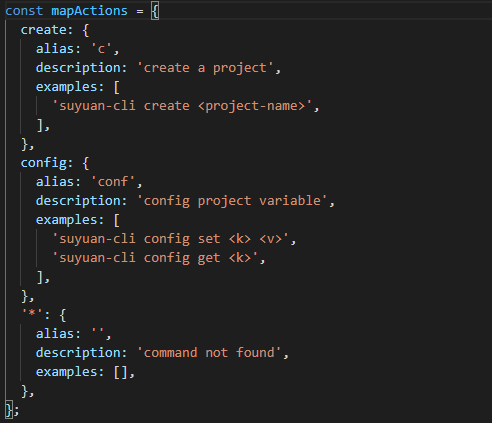
\*alias //命令别名

\*description //命令描述

\*action //命令匹配成功后执行动作



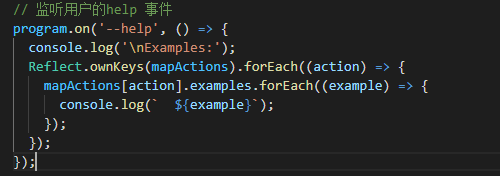
通过配置一个变量来循环执行注册命令：



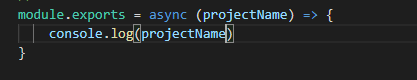
循环这个变量：



添加--help事件：输出example



require(path.resolve(\_\_dirname, action))(...process.argv.slice(3)); 这句的意思是通过action名字加载对应的模块，执行模块返回的函数，入参为第三个参数，例如在cmd输入命令suyuan-cli create my-project的时候，会调用create.js,并且会传入参数my-project，新建目录src==>create.js 输入下面代码



在cmd下输入suyuan-cli create my-project 测试输出：



1. 拉取git项目模板

安装npm i axios -D，在create.js配置下面代码：

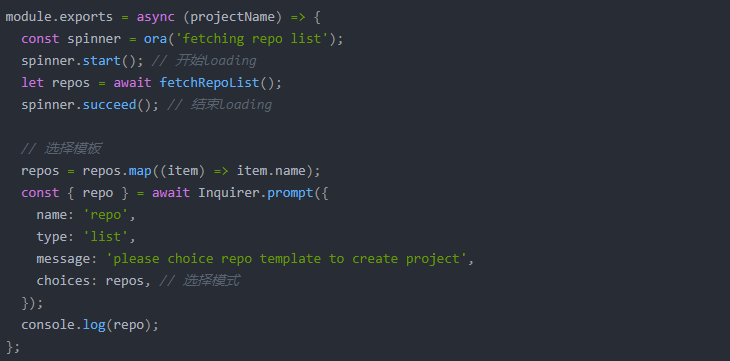


输入 suyuan-cli create myproject，得到模板名称：

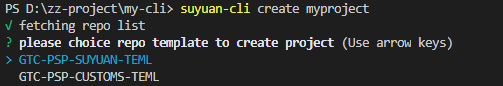


1. 拉去模板的时候加上加载提示，选择提示，安装 inquirer & ora， cnpm i inquirer ora -D，

\*inquirer 用于提示loading，ora用于选择



输入命令suyuan-cli create my-project: 可以选择



* 1. 封装loading函数方便调用：

// 封装loading效果

const waitFnloading = (fn, message) => async (...args) => {

    const spinner = ora(message);

    spinner.start();

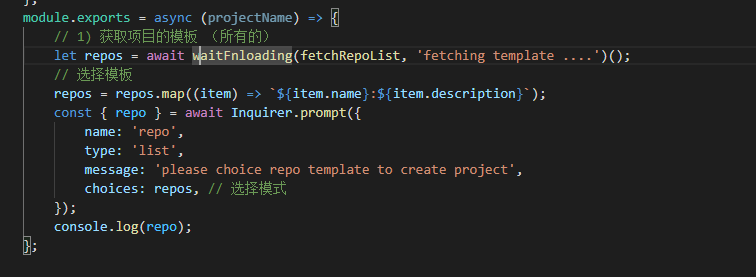
    const result = await fn(...args);

    spinner.succeed();

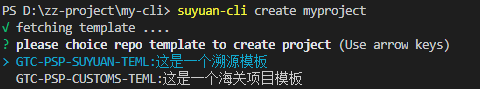
    return result;

};

* 1. 具体的调用方法：



测试输出：



* 1. 拉取每个项目tags，定义拉取函数

// 抓取tag列表

const fechTagList = async (repo) => {

    const { data } = await axios.get(`https://api.github.com/repos/suyuan-cli/${repo}/tags`);

    return data;

};

拉取并且选择：

    //用模板名称加载tags

    let tags = await waitFnloading(fechTagList, 'fetching tags ....')(tempName);

    tags = tags.map((item) => item.name);

    //选择tags

    const { tag } = await Inquirer.prompt({

        name: 'tag', // 获取选择后的结果

        type: 'list',

        message: '请选择版本号',

        choices: tags,

    });

* 1. 根据模板名称和tags下载模板

//下载函数

const download = async (repo, tag) => {

    let api = `suyuan-cli/${repo}`;

    if (tag) {

        api += `#${tag}`;

    }

    // /user/xxxx/.template/repo

    const dest = `${downloadDirectory}/${repo}`;

    await downloadGitReop(api, dest);

    return dest; // 下载的最终目录

};

    //根据模板名称和tags下载模板

    // download-git-repo

    const result = await waitFnloading(download, 'download template')(tempName, tag);

* 1. 如果项目模板中没有ask.js，就直接copy项目到当前的运行目里面

  //如果项目模板中没有ask.js，就直接copy项目到当前的运行目里面

    if (!fs.existsSync(path.join(result, 'ask.js'))) {

        await ncp(result, path.resolve(projectName));

        console.log("cd " + projectName + " and run npm install")

    }

* 1. 如果有ask.js 代表是要询问配置信息的，这里用到的原理就是变量ask.js里面的问题

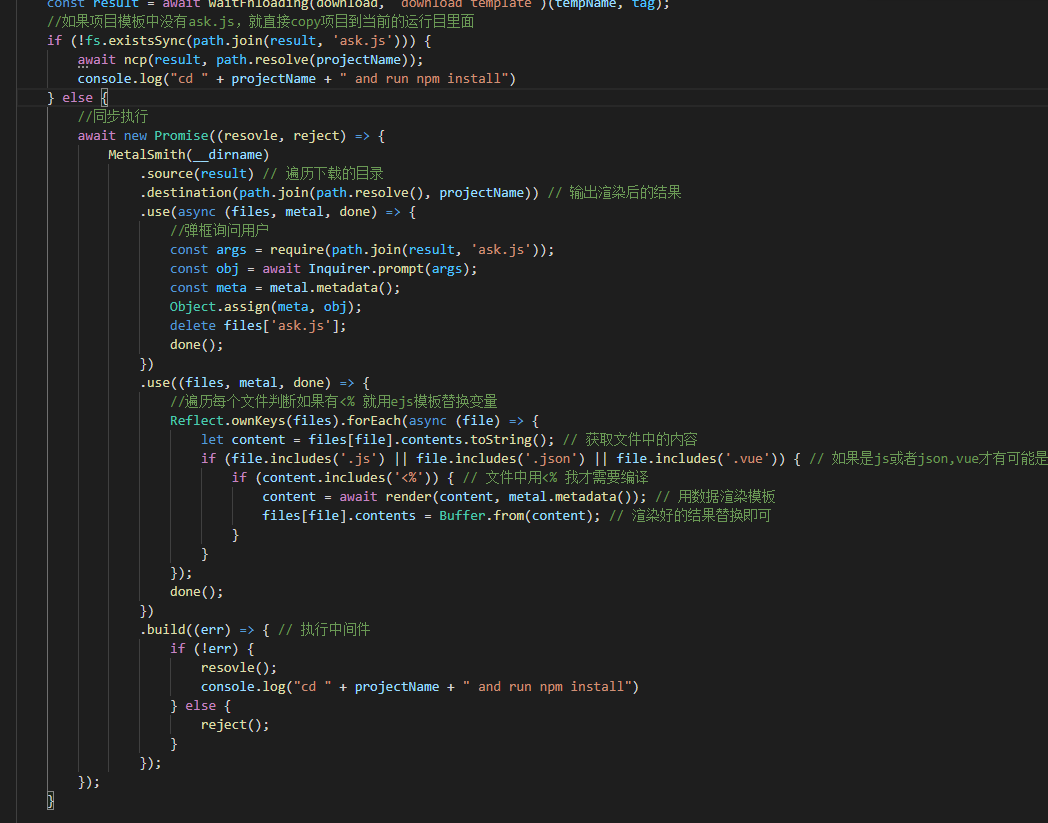
得出用户配置对象，然后遍历项目中的所有文件，遇到<%开头的就用模板变量替换

安装下面模块：

cnpm i metalsmith ejs consolidate

metalsmith 是遍历文件用的，ejs是模板引擎，consolidate是集合多种模板输出函数的模块

具体代码解析如下：



至此，输入shuyuan-cli 就得到下面的输出流程：

