1.node、express安装及项目生成

安装完node后，进入目标文件夹，使用npm安装express：npm install -g express --save；

npm运行express生成项目，express myApp（任意起的名）；

进入myApp目录，npm运行npm install，生成node\_modules文件夹，至此项目生成完毕。

2.运行项目环境

npm运行node app,启动服务器，根据app.js中的路由配置打开对应路径文件。

3.package包

js模块的基本单位是单个js文件，但复杂模块需要多个子模块组成，这种多个子模块组成的大模块成为package，每个包都有一个入口文件（其他文件均通过该文件加载），

如NODE\_PATH='home/user/lib'，加载home/user/lib下的cat模块，其中cat文件夹中含三个文件：head.js、body.js、main.js,入口文件为main.js，①则可以写成require('cat/main');

②该方法用到了入口文件名，不能体现包的整体性，可以在cat包下建index.js作为入口文件，则写为require('cat')即可默认加载其中的index文件；③在cat目录下新建package.json配置文件，配置包信息，如{"name": "cat", "main": "main.js"}, NodeJS会根据包目录下的package.json找到入口文件的位置。

4.node.js文件操作

例：(1)小文件拷贝

var fs = require('fs');

function copy(src, dst) {

fs.writeFileSync(dst, fs.readFileSync(src));

}

function main(argv) {

copy(argv[0], argv[1]);

}

main(process.argv.slice(2));

注：fs为node内置文件操作模块；

process是一个全局变量，可通过process.argv获得命令行参数，argv[0]固定等于执行程序的绝对路径，argv[1]固定等于主模块绝对路径，因此第一个命令行参数是从argv[2]开始的，如命令行运行node ./hello para，则argv=['node', 'path/to/hello', 'para']

(2)大文件拷贝

var fs = require('fs');

function copy(src, dst) {

fs.createReadStream(src).pipe(fs.createWriteStream(dst));

}

function main(argv) {

copy(argv[0], argv[1]);

}

main(process.argv.slice(2));