使用说明

该工具内部集成了本地开发服务器、前端模块化开发管理功能、动态加载模板、sass编译压缩、js合并压缩等功能。

此次升级移除了用于自动刷新的BrowserSync，添加webpack用于自动刷新。主要是因为BrowserSync自动刷新时很多时候都会比较慢，影响工作效率。

1. 安装

首先需要安装node.js

1. 全局安装webpack及webpack-dev-server，如不需要自动刷新（即运行guilin server命令），这步可以省略。命令：

npm install -g webpack

npm install -g webpack-dev-server

1. 添加phantomjs为系统变量环境（下载地址：http://phantomjs.org/）

下载并解压到任意目录如D:\phantomjs\phantomjs.exe，添加环境变量，我的电脑－>属性－>系统高级设置->环境变量－>系统变量里找到path，追加到后面。如果不需要动态加载模块，这步也可以省略。

1. 将文件夹guilin复制到以下路径C:\Users\当前用户名\AppData\Roaming\npm\ node\_modules，

然后进入C:\Users\当前用户名\AppData\Roaming\npm\node\_modules\guilin目录，执行npm install命令安装依懒。

4、最重要的一点：将目录下的guilin.cmd复制到以下

目录C:\Users\当前用户名\AppData\Roaming\npm，才能使用下面的所有命令。

目录下包含了static文件夹，guilin init初始化时会将目录下的文件复制到当前项目目录，因此可以将一些常用的脚本样式图片等放到对应的目录去，初始化时就不用再次复制了。

1. 使用

在适合的地方新建文件夹，例如test。然后在此目录下执行项目初始化命令：

1、guilin init

这条命令会在项目目录在创建基本目录结构：

Test

-- build 打包发布后的目录

--src 发开源文件目录

--css

-- images

--js

--sass

--model 存放公共模块，如header.html

--webpack webpack配置相关文件，如不需要自动刷新可删除

-- package.json 项目配置信息

2、guilin build

简写为guilin b

此命令可将src下的文件编译并复制到指定目录下如test，引入模板带有参数时，可能需要执行两次打包命令。

3、guilin server

简写为 guilin s

　 启动服务，执行后可在浏览器输入<http://localhost:8080>查看页面，修改了源文件会自动刷新页面。

4、guilin watch

简写为 guilin w

　 开启监听命令，类似于server，不同的在于它不会启动http服务。

5、guilin merge

简写 guilin m

根据配置合并并压缩js，在打包时会执行此命令，建议build执行完后再独立执行合并压缩。

6、guilin imgtobase64

简写 guilin t

将css样式中的背景图片转为base64。需要在配置中设置imgToBase64为true。此命令没有与其它命令关联，需要转换图片时要单独执行才能转换。

1. Package.json

一般情况无需配置，用默认的即可。

dist: 编译生成目录，默认为空即为根目录test，如将文件打包到目录build，dist:”build”

port: 启动服务时的端口

serverIp: 用于访问的本机ip地址，开启server命令有效，不填写时只能通过<http://localhost>来访问。

outputStype: sass输出类型,可选nested，expanded，compact，compressed(默认)

autoPreFixer:添加css3兼容前缀，默认为true

uglifyjs:压缩js，在输入目录生成files.min.js，默认为false

mergeJs:合并js,用+号连接，如a.js+b.js

mergeJsName:合并后的js输出文件名

imgToBase64:将样式中的背景图片转为base64，默认为false

imgTobaseDel:图片转为base64后是否将图片删除，多个样式文件同时引用同一图片时，后面的图片会转换失败，此时建议设为false

1. 模块化开发

对于一个页面，除自身的html模板，样式和脚本外，同时还包括如header，footer等公共的部分，要使用一个公共header很简单，只需在页面引用模块html即可，如model下新建header.html。内容为：

<div class=”header”>这是公共头部</div>

在src下新建index.html，内容为：

<body>

*<!-- include href="header.html" -->*

<p>这里是当前页面内容</p>

</body>

如果已经开启了server或是watch命令，此时src目录下会生成index.html文件，内容为：即已把头部内容包括进来了

<body>

<div class=”header”>这里是公共头部</div>

<p>这里是当前页面内容</p>

</body>

1. 动态内容生成

很多时候header.html也是需要针对不同的页面有一些小改变，例如菜单导航显示当前位置。加载动态时只需要传参数即可，然后在公共模板如header.html里接收参数，并作相关的处理，这里加载模板并解释js需要安装phantomjs。它需要一个url将参数传过去。

引入动态模板如

*<!-- include href="header.html?a=1" -->*

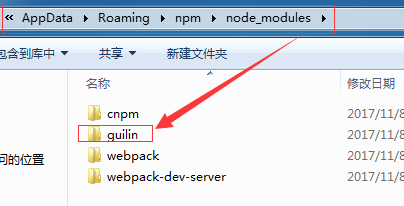
然后在header.html里接收参数a。

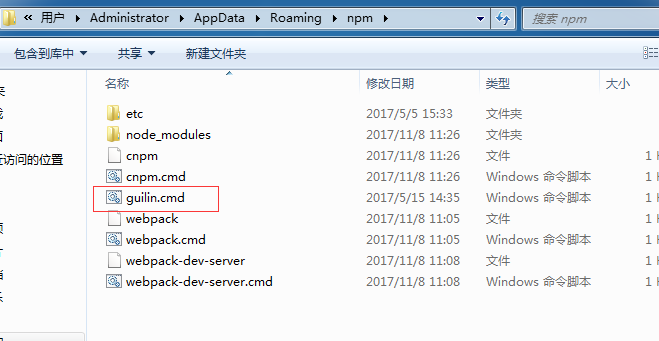
这里有个特殊的地方，因为是url访问，header.html需要一个完整的html页面结构，即包括html,head,body这些标签，否则访问到的有可能是乱码，因此带有参数访问时，只会返回header.html里面id=”page”的内容，其它的不会被返回

六、特别注意

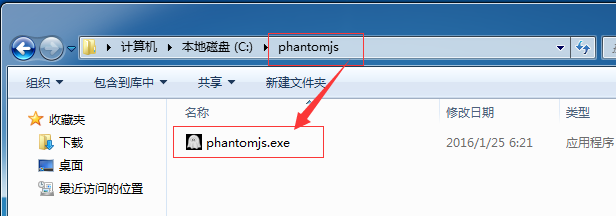
model下的文件修改是不会重新编译的，需要重新打包，若引用模板时带有参数，需要build两次

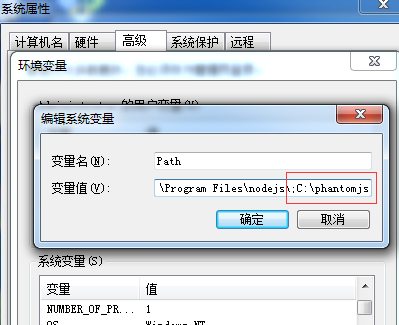
使用图解



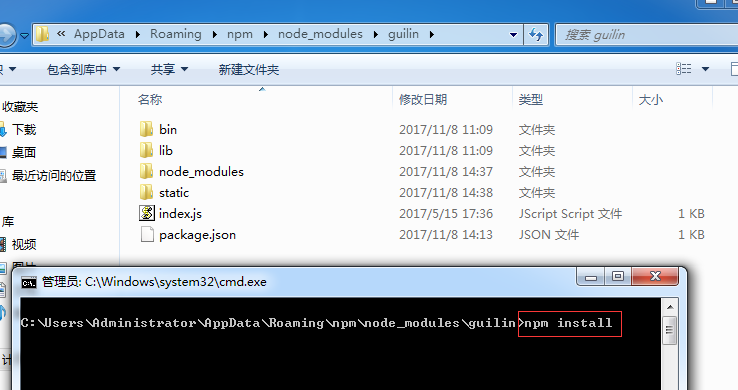


复制到指定路径

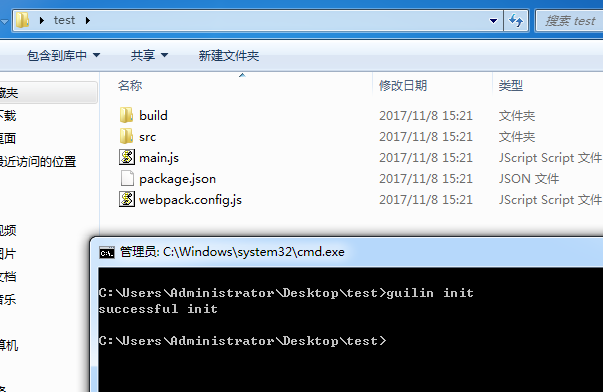




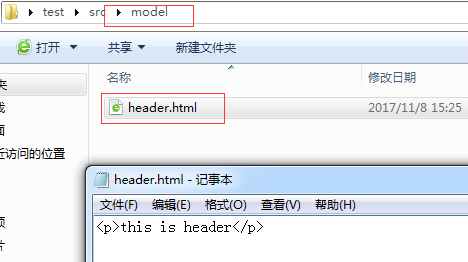
添加环境变量



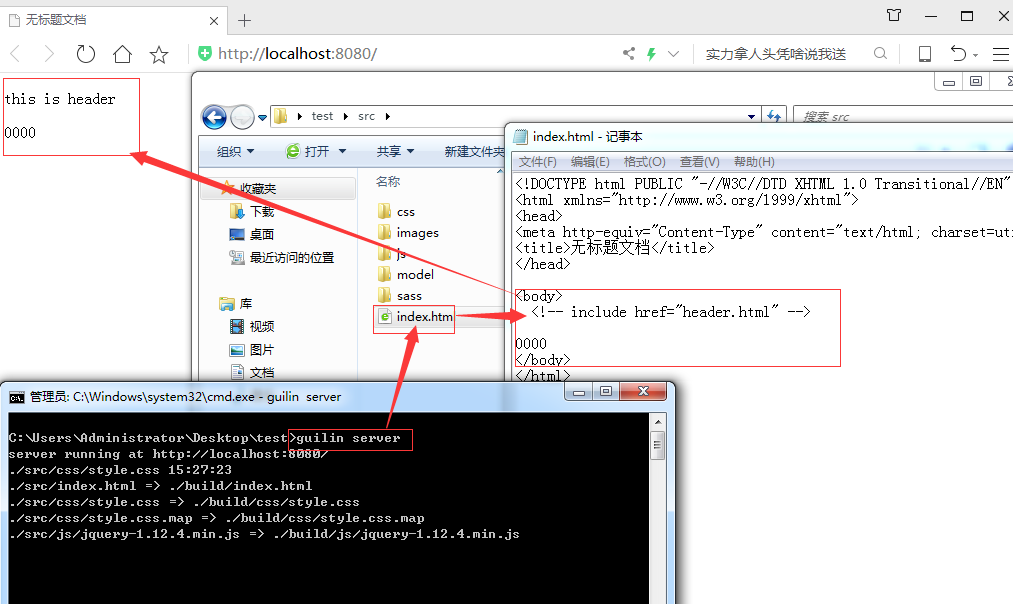
运行npm安装命令



初始化



公共头部模块



运行命令guilin server，自动弹出浏览器窗口；

修改src下的index.html，浏览器自动刷新。