**安装**

fis-plus 的 自动化/辅助开发工具 被发布为一套 npm包，它对环境的要求是：

* 操作系统：任何能安装 nodejs 的操作系统
* node版本：>= v0.8.0
* jre版本：>= v1.5.0 【如果不需要本地调试服务器，可以忽略java环境要求】
* php-cgi版本：>= v5.0.0 【如果不需要本地调试服务器，可以忽略php-cgi环境要求】

**安装nodejs**

* [安装nodejs](http://nodejs.org/)

**npm**

[npm](https://www.npmjs.org/)是nodejs的包管理工具。安装nodejs后，npm就自动一起安装了。

* 用nodejs写的模块都发布在npm上。[npm网站](https://www.npmjs.org/)
* 用户需要使用npm install命令来安装nodejs模块。更多npm使用，执行 npm -h 来查看
* 由于npm经常被墙，安装fis的时候会出现速度过慢，或者安装不上的问题 。
* 可以通过 npm的 --registry 参数指定仓库。指定国内的npm镜像来解决npm被墙的问题。

例如：

npm install <some npm module> -g --registry**=**镜像

* 下面提供一个国内镜像
  + --registry=[http://r.cnpmjs.org](http://r.cnpmjs.org/)
* 百度内部可以使用公司内镜像
  + --registry=[http://npm.internal.baidu.com](http://npm.internal.baidu.com/)

**安装 fis-plus**

nodejs安装好后，命令行执行

npm install -g fis-plus

安装好fis-plus之后，执行 fisp -v，如果能看到以下信息，则表明安装成功。*如果安装过程中遇到什么问题，可以到*[*https://github.com/fex-team/fis-plus/issues?state=open*](https://github.com/fex-team/fis-plus/issues?state=open)*提问题，或者QQ询问。*

➜ ~ fisp -v

v0.7.2

\_\_/\\\\\\\\\\\\\\\\_\_/\\\\\\\\\\\\_\_\_\_\_/\\\\\\\\\\\\_\_\_

\_\/\\\///////////\_\_\/////\\\///\_\_\_\_/\\\/////////\\\\_

\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\/\\\\_\_\_\_\_\_\//\\\\_\_\_\_\_\_\///\_\_

\_\/\\\\\\\\\\\\_\_\_\_\_\_\_\_\_\/\\\\_\_\_\_\_\_\_\////\\\\_\_\_\_\_\_\_\_\_

\_\/\\\///////\_\_\_\_\_\_\_\_\_\_\/\\\\_\_\_\_\_\_\_\_\_\_\////\\\\_\_\_\_\_\_

\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\////\\\\_\_\_

\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\/\\\\_\_\_\_\_\_/\\\\_\_\_\_\_\_\//\\\\_\_

\_\/\\\\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\\\\\\\\\\\\_\///\\\\\\\\\\\/\_\_\_

\_\///\_\_\_\_\_\_\_\_\_\_\_\_\_\_\///////////\_\_\_\_\///////////\_\_\_\_\_

**安装lights**

[lights](http://lights.baidu.com/)是fis提供的包管理工具，托管了fis所有资源。是使用fis的时候，必不可少的利器。

npm install -g lights

**安装Java**

* [安装java](http://java.com/)

**安装php-cgi**

* [mac安装](https://gist.github.com/xiangshouding/9359739)

mac下安装php-cgi有多种方法，这里只介绍比较简单的两个方法；

* + 用brew安装
  + 直接下载安装XAMPP

**用brew安装**

如果安装了xcode，那么推荐使用brew来安装php。详细使用方法见[官网](http://brew.sh/)，这里只说明如何装php-cgi；

$ brew install php55 --with-cgi

如果安装提示没有php55，请用 brew tap homebrew/homebrew-php 后再安装

如上，方法很简单，等安装成功后即可使用；

**直接下载安装XAMPP**

到[XAMPP官网](http://www.apachefriends.org/)下载Mac版本，双击安装；等安装成功后需要把XAMPP的bin目录设置到 环境变量里面； \* 使用zsh

$ echo 'export PATH=/Applications/XAMPP/bin:$PATH' >> ~/.zshrc

$ source ~/.zshrc

* + 使用bash

$ echo 'export PATH=/Applications/XAMPP/bin:$PATH' >> ~/.bashrc

$ source ~/.bashrc

* [windows安装](https://gist.github.com/lily-zhangying/9295c5221fa29d429d52)

windows安装方法很多，下面介绍最简单的一种。

**直接安装xampp**

地址：[<http://www.apachefriends.org/zh_cn/index.html>]

* + 下载xampp并安装，将xampp/php路径加入环境变量中。
  + 就安装了php和php-cgi。
  + cmd命令行输入php-cgi -v ，就看到php-cgi版本号。php-cgi就装好啦
* [linux安装](https://github.com/fouber/install-php-cgi-1)

**示例**

*以下示例都是在命令行下操作的，如果你是windows用户，请打命令的时候忽略命令前的$，而且请打开cmd来执行这些操作*

已经准备好了一个fis-plus的前端项目，只需要经过以下四步，就可以完整运行这个项目，并看到结果。

* 初始化本地模拟环境
* 下载Demo
* 发布
* 预览

**初始化本地模拟环境**

为了前后端开发分离，来并行开发，fis-plus提供了一套本地环境模拟的工具，安装并初始化它后就能方便的模拟线上环境了。

$ fisp server init

**下载Demo**

FIS的所有示例及其组件都用包管理工具lights进行管理，使用lights安装demo。

$ lights install pc-demo

其实fisp已经集成了lights的客户端。

和上面等值的用法；

$ fisp install pc-demo

**发布**

$ cd pc-demo

$ fisp release -r common

$ fisp release -r home

**预览**

$ fisp server start *#启动服务器*

启动服务的时候，启动成功后会自动打开浏览器，访问首页，这时候你应该打开demo首页，并和下图是一致的。



**示例解说**

自此，一个前端项目已经运行起来了。你可以看一下pc-demo的源码，其中包含两个模块

* common
* home

模块这个词会贯穿整个文档，以及整个fis-plus的使用。为什么会有模块这种东西？ 当前端代码很多时，不得不面临分组件，分页面。为了发布迭代方便，不得不把它们分为不同的子系统。比如用户信息、首页、详情页等等。

模块就是一个子系统，而在fis项目中用namespace和fis-conf.js来区分。每一个模块会有一个配置文件fis-conf.js，还会取名不同的namespace。这主要是为了区分模块之前的静态资源。

继续进入home模块，可以看看有几个目录及其文件

.

├── fis-conf.js

├── page

├── server.conf

├── static

├── test

└── widget

无疑，这就是使用fis-plus需要遵循的目录规范，为什么要有目录规范，可能在网上可以找到很多答案，这里就不再赘述。

说一下这几个目录所代表的意思；

* page 页面模板
* widget 组件，模板组件，JS组件，CSS组件，会被组件化
* static 这个目录下放一些不需要组件化的公共库，比如lazyload.js
* test 放置一些测试数据，和page下的模板相对应，表明哪个模板用哪个数据文件进行渲染
* server.conf 这是一个很有用的文件，它里面可以配置url转发，可以方便在本地模拟ajax请求等。

*细心的你有可能发现了一个比较专业的词汇****组件化****，组件化的细节比较繁多，准备新开一节说明*

**page 如何引入 widget**

* 如示例中index.tpl如何应用widget/header/header.tpl ? 如果用过smarty的你可能会想到include，但在FIS-PLUS中引入widget有个跟include类似的插件完成，widget。
* {%widget name="home:widget/header/header.tpl"%}

home:widget/header/header.tpl 这个是FIS中[静态资源id](http://oak.baidu.com/docs/fis-plus/user/quickstart.html#%E9%9D%99%E6%80%81%E8%B5%84%E6%BA%90id)

**模板中如何使用widget目录下的静态资源**

根据目录规范widget目录下的js讲被进行[组件化封装](http://oak.baidu.com/docs/fis-plus/user/quickstart.html)，根据[同名依赖](http://oak.baidu.com/docs/fis-plus/user/quickstart.html)原则；

* 当使用某个widget下模板是，同名js,css将会被加载。
* 当使用某个widget下的js时，其同名css会被加载。

**加载JS**

由于js进行了组件化封装，比如通过require或者require.async函数来执行其中逻辑。

{%script%}

require('/widget/a.js');

{%/script%}

如上，在模板中使用widget下的js，必须放到{%script%} {%/script%}之间，用它来代替js的内联用法。

**加载css**

说完加载js的方法，css如何引入呢？

* 同名依赖，被依赖
* 通过smarty的require插件

{%require name="home:widget/a.css"%}

**静态资源id**

静态资源id会贯穿整个用户文档，跟使用密切相关，所以它很重要。现在需要弄清楚两件事情

* 静态资源id是如何在fis-plus中计算的？
* 静态资源id在那些情况下使用，那些情况下必须用静态资源id？

**静态资源id是如何在fis-plus中计算的？**

fis-plus必须要指定模块的namespace，所以静态资源id被标记为

namespace:<资源相对于模块根目录的路径>

比如:

* home/static/a.js home目录为模块跟目录，home为namespace，则静态资源id就为 home:static/a.js
  + namespace可以为任意值，可以不跟模块根目录相同。

**静态资源id在那些情况下使用，那些情况下必须用静态资源id？**

* 使用widget、require、html等**smarty**插件时，必须指定资源的id
* require、require.async等**JavaScript**函数，可以使用id