|  |
| --- |
|  |
| Steel框架使用说明 |
|  |
|  |
| **撰写人：谢飞xiefei2 / 王政wangzheng4** |
| **日期：2015/01/12­­­­­­­­­** |

|  |
| --- |
|  |

目 录

[**一、steel介绍：** 5](#_Toc433986574)

[**二、steel使用规范：** 6](#_Toc433986575)

[**2.1 后端主页规范：** 6](#_Toc433986576)

[**2.2 前端开发约定：** 8](#_Toc433986577)

[**2.2.1 工程目录结构：**(按字母顺序) 8](#_Toc433986578)

[**2.2.2 文件依赖：** 9](#_Toc433986579)

[**2.2.3 配置：** 9](#_Toc433986580)

[**2.2.4 模板开发约定：** 10](#_Toc433986581)

[**2.2.5 模块化分** 10](#_Toc433986582)

[**2.2.6 渲染** 10](#_Toc433986583)

[**2.2.7 切换页面与获取url数据：** 12](#_Toc433986584)

[**2.2.8 参数说明：** 12](#_Toc433986585)

[**2.2.9 特殊情况的处理办法** 13](#_Toc433986586)

[**2.2.10 css约定与模板的使用** 13](#_Toc433986587)

[**2.3 前端安装、调试、上线 (gulp工具项目构建)** 14](#_Toc433986588)

[**2.3.1 环境说明** 14](#_Toc433986589)

[**2.3.2 安装** 14](#_Toc433986590)

[**2.3.3 使用服务** 14](#_Toc433986591)

[**2.3.4 server\_back目录与server\_front目录** 15](#_Toc433986592)

[**2.3.5 host与gulpfile配置** 15](#_Toc433986593)

[**2.3.6 上线** 16](#_Toc433986594)

[**附录：** 16](#_Toc433986595)

文档说明：

编写者：

创建时间：2015-01-12

当前版本号：V 1.0

面向对象：前端开发者

内容说明：描述如何使用steel框架

修改时间：2015-08-06

**一、steel介绍：**

    steel是一个致力于分离前后端职责的前端框架，它的思想是通过在浏览器与服务器之间共享网页渲染逻辑和一部分业务逻辑来实现跨浏览器、服务器工作。

**二、steel使用规范：**

**2.1 后端主页规范：**

静态页，形如：

        html

                head

                        meta(charset='utf-8')

                        title Hello World

                body

                script(src="http://js.t.sinajs.cn/SteelHelloWorld/src/js/lib/lib.js")

                script(src="http://js.t.sinajs.cn/SteelHelloWorld/src/js/app.js")

                script.

                        steel.boot('app');

对应后端EPF框架所使用的phtml文件，后短发输出如下：

<!DOCTYPE html>

<html lang="en">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0, minimum-scale=1.0, maximum-scale=1.0" />

<title>粉丝服务平台</title>

<link rel="stylesheet" href="http://img.t.sinajs.cn/t4/appstyle/e/apps/message/css/mobile\_base.css?version=<?= htmlentities($g\_version\_css, ENT\_QUOTES, 'UTF-8') ?>" />

<script src="http://js.t.sinajs.cn/t6/apps/fans\_service\_mobile/js/lib/lib.js?version=<?= htmlentities($g\_version\_css, ENT\_QUOTES, 'UTF-8') ?>"></script>

<script src="http://js.t.sinajs.cn/t6/apps/fans\_service\_mobile/js/app/groupmsg.js?version=<?= htmlentities($g\_version\_css, ENT\_QUOTES, 'UTF-8') ?>"></script>

<?php include(dirname(\_\_FILE\_\_) . '/../../common/include/config.phtml') ?>

<script>steel.boot('app/groupmsg');</script>

</head>

<body></body>

</html>

**2.2 前端开发约定：**

**2.2.1 工程目录结构：**(按字母顺序)

            | build    /\*上线目录，用于推静态池，gulp build命令生成\*/

| doc /\*文档目录，用来存放数据约定/接口文档等\*/

            | node\_modules    /\*依赖的插件目录\*/

| server\_front    /\*debug目录，有gulp server 命令可生成\*/

 | server\_back    /\*手动书写，用来模拟后端接口和路由文件\*/

            | src    /\*源码目录\*/

                    | \_html    /\*demo目录，存放纯静态的html文件,\*/

                    | \_psd    /\*用来存放psd文件，下划线目录文件不会带进仿真环境和上线环境\*/

| css    /\*样式源码\*/

| img   /\*图片目录\*/

                    | js    /\*逻辑源码\*/

                           | app    /\*用来存放应用配置文件\*/

 | common   /\*用来存放公用模块或组件\*/

                            | comp    /\*用来存放各个模块（组件）\*/

| example    /\*模块目录，例如名字叫example\*/

controller.js /\*模块控制器\*/

logic.js /\*模块逻辑\*/

tpl    /\*模块模板\*/

| children   /\*example的子模块目录，例如叫做children\*/

controller.js /\*子模块控制器\*/

logic.js /\*子模块逻辑\*/

tpl    /\*子模块模板\*/

             | lib.js    /\*框架类库文件，gulp会合并该目录为一个文件叫lib.js\*/

steel.js /\*框架文件\*/

jquery.js /\*类库文件，或使用STK等其他库\*/

debugConfig.js /\*debug状态配置\*/

            |Gruntfile.js    /\*grunt配置文件\*/

            |package.json    /\*nodejs模块声明文件，用于会编辑模块依赖部分\*/

**2.2.2 文件依赖：**

steel使用require、module.exports、exports的方式处理加载依赖，使用方法同nodejs的require、module.exports、exports一致。

**2.2.3 配置：**

在文件app.js里对文件进行配置，例如：

 /\*app.js 内容  START\*/

            steel.config({

singlePage: fasle, //是否单页面，决定了a链接等的跳转与否，默认是true

                    basePath: '/',    //若jspath与csspath没有配置自己的路径，则使用此路径

                    jsPath: '/js/',    //配置jsPath基础路径

                    cssPath: '/css/',    //配置css基础路径

                    ajaxPath: '/',    //配置ajax基础路径

                    mainBox: document.body,    //渲染所需的最外层id或节点

                    router: [    //配置路由信息

                            ['/', 'controller/helloworld'],

                            ['/index.html', 'controller/helloworld'],

                            ['/html/index.html', 'null'],

                            ['/html/', 'controller/helloworld'],

                            ['/html/helloworld2.html', 'controller/helloworld2']

                    ]

            });

            /\*app.js 内容  END\*/

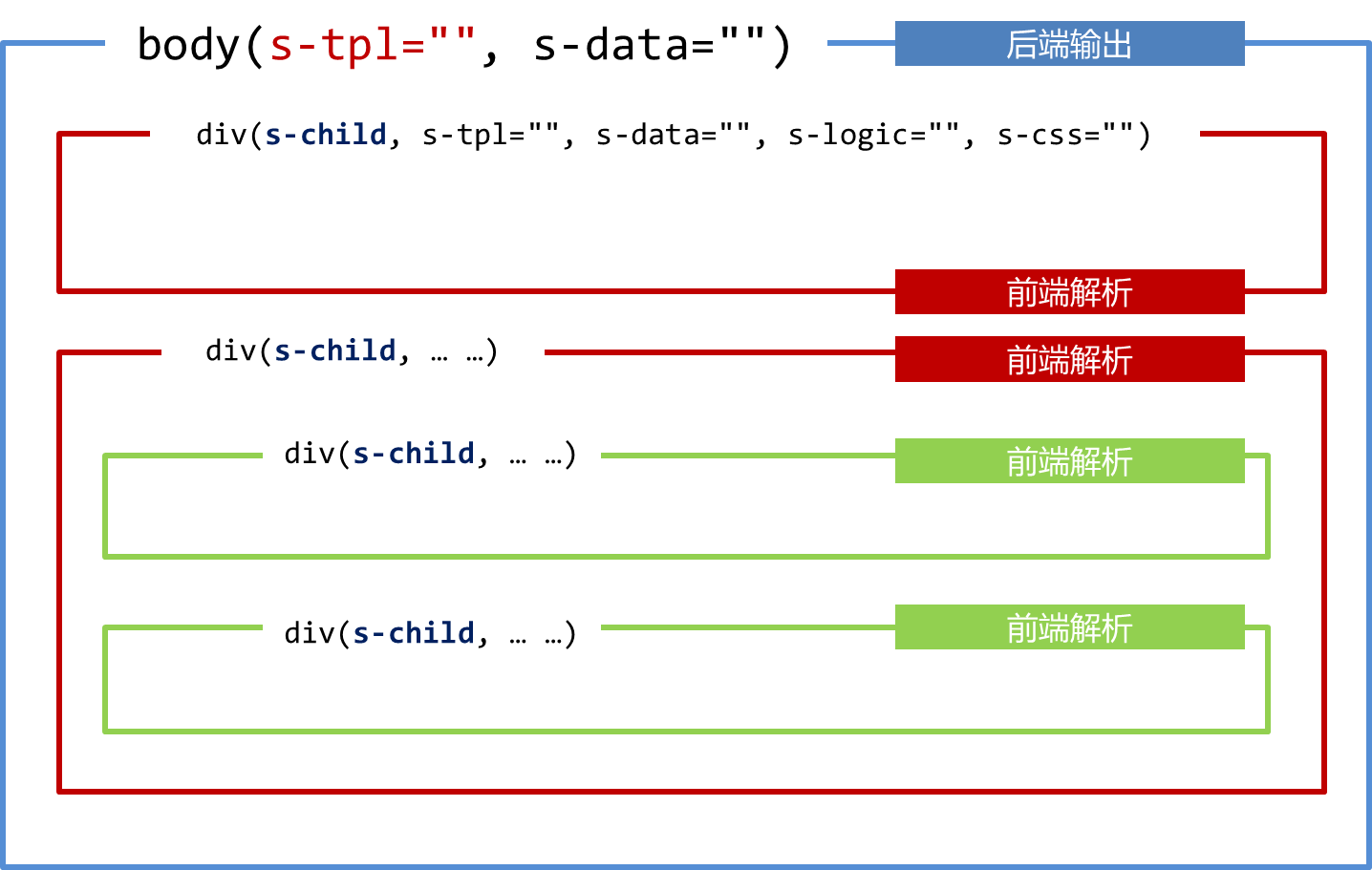
**2.2.4 模板开发约定：**

前端模板文件统一由src/js/tpl目录管理

模板使用jade开发

jade模板编写中文规范：<http://segmentfault.com/a/1190000000357534> 或 jade官网 <http://jade-lang.com/>

**2.2.5 模块化分**



**2.2.6 渲染**

**2.2.6.1 第一种方式：**节点渲染

**【**最外层模块】：（最外层一般是指body）

body(s-tpl="tpl/helloworld", s-data="/helloworld")

约定：

* 节点由config中的mainBox属性获得。
* 可以配置的模块资源是s-tpl、 s-data、s-css、s-logic，不必配置s-controller属性。
* 在没有controller配置信息的情况下，渲染必须配置的条件是存在s-tpl与s-data，其中s-data为空时，其值标识为null，否则不能渲染。
* 必须配置路由信息，第二个参数可以为null，此时按节点所给的属性资源进行渲染。

**【**子模块】：

div(s-child, s-tpl="tpl/child", s-data="null", s-css="css/child", s-logic="logic/child", s-controller="controller/child" )

约定：

* 需要标识条件属性s-child，没有标识s-child属性的将不被视为子模块。
* 可以配置的模块资源是s-child(必选)、s-tpl、s-css、s-logic、s-data、s-controller
* 在没有controller配置信息的情况下，渲染必须配置的条件是存在s-tpl与s-data，其中s-data为空时，其值标识为null，否则不能渲染。

**2.2.6.2 第二种方式：**controller控制器控制渲染

【资源的配置方式】

css: 'css/helloworld'

tpl: 'tpl/helloworld' 或 require('tpl/helloworld');

data: '/helloworld' 或 {helloworld: 'hello'}

logic: 'logic/helloworld' 或 require('logic/helloworld');

【controller的使用】

control.set('css', 'css/helloworld');

control.set('tpl', 'tpl/helloworld');  或 control.set('tpl', require(''tpl/helloworld''));

control.set('data', '/helloworld'); 或 control.set('data', {type:'hellow'});

control.set('logic', require('logic/helloworld'));  或  control.set('logic', 'logic/helloworld');

【s-child与s-controller】

若s-child有值，则优先使用s-child的值作为controller配置与control.set('children', 'xx')

**情况一：**

div(s-child="a", s-controller="controller/helloworld2")

control.set('children', {a: 'controller/helloworld1'});//此种情况下**controller**/**helloworld1生效**

**情况二：**

div(s-child, s-controller="controller/helloworld2")//此种情况下**controller**/**helloworld1生效**

**2.2.7 切换页面与获取url数据：**

steel.router.set(url); //切换

steel.router.get() 当前页url数据资源获取

**2.2.8 参数说明：**

* logic的参数node：是指当前模块的box节点
* controller的两个参数：

control：{

set:function(type, res);//css, data, logic,

}

* rootScope//用于传递数据，可以把数据赋值给此变量，在父/子的controller里使用。

**2.2.9 特殊情况的处理办法**

           问：两个页面使用相同的模块，怎么破？

                答：只需要写一份，同时include一个模块。

                问：我在logic文件怎么调用controller方法？

                答：不可以，两个文件各司其职，互不干扰。

**2.2.10 css约定与模板的使用**

a. 模块的css需要有结束标识42px，例如:

当前模块controller.set({css: 'customMenu/main'})，那么在main.css这个文件的最底部需要写成： #S\_CSS\_customMenu\_main{ 42px ; }，这个id就是S\_CSS\_前缀加上所set的css把斜杠变成下划线后拼接而成的，用来表示css文件加载完成。

b. 模板jade文件有两处使用，除了被control.set({tpl: 'tpl/customMenu/main'})用来渲染外，其他需要用到html模板的地方也可以通过require的办法使用，例如：

var tplFn = require('tpl/customMenu/main');

var html = tplFn(data);

此方法可以取到所需要的动态模板。

**2.3 前端安装、调试、上线 (gulp工具项目构建)**

**2.3.1 环境说明**

gulp版本的工具是grunt构建工具的改进版，性能更高，提升了开发效率

该集成开发工具集成了多种处理任务，这些处理任务依赖的环境如下：

        nodejs、gulp

**2.3.2 安装**

Nodejs ：请参考http://nodejs.org/ v0.10+

gulp ：npm install -g gulp

nodejs deps ：npm install

**2.3.3 使用服务**

默认端口80

可使用的命令：

**\*gulp debug\***：调试处理：对src目录文件进行debug处理，生成调试代码，包括模板处理、脚本wrap和合并、静态文件copy等

**\*gulp dist\***：仿真处理：对src目录文件进行dist处理，生成仿真代码，除做debug中的处理外，还有css压缩合并

**\*gulp build\***：上线处理：生成上线文件,会把可上线的结果处理了build目录下

**\*gulp server\***：启动调试服务器命令 --dist 为仿真服务器 --pm2 为使服务后台运行(win下无效)');

**\*gulp serverStop\***：关闭服务器命令 当存在后台服务时有效

**2.3.4 server\_back目录与server\_front目录**

server\_back目录：用来模拟后端的文件放置目录（路由和接口）

分为aj和非aj两种目录。aj目录用来存放假接口与使用grunt （另一种工程构建方式）时的假接口目录作用一样；非aj目录用来模拟后端的路由及后端输出。

server\_front目录：debug目录，用来存放静态文件

工具自动生成，与使用grunt时的debug目录作用一样

生成来源：src目录

**2.3.5 host与gulpfile配置**

**host配置 ——**

127.0.0.1 js.t.sinajs.cn js1.t.sinajs.cn js2.t.sinajs.cn

127.0.0.1 e1.weibo.com #模拟后端host，跟gulpfile的back\_hostname值一致

10.73.15.236 e.weibo.com #php host

#输入e1.weibo.com/v1/public/ custommenu/main 走的是前端模拟目录文件和aj数据

#此时输入e.weibo.com/v1/public/custommenu/main 走的是后端真实目录文件和数据

**gulpfile配置 ——**

port：服务端口

pathnamePrefix：'/t6/apps/fans\_service\_mobile/', //工程path

front\_base：'server\_front', //debug目录名

back\_base：'server\_back', //模拟后端的文件放置目录

front\_hostname：'js.t.sinajs.cn img.t.sinajs.cn', //静态资源host配置，与host

back\_hostname ： 'e.weibo.com e1.weibo.com' //后端的host，目的是模拟后端的页面路由请求，提供前端可仿真的功能，比如 /index 对应 /html/index.html

**2.3.6 上线**

上线前使用gulp build命令

QB上线

注意：

1. QB的线上目录是从svn的build目录获取，发布系统上“输入发布代码路径”，形如：https://svn1.intra.sina.com.cn/weibo/ria/t6/apps/enp\_mindPage /trunk/build/
2. QB填入项“输入标签路径”，指存放发布版本代码的目录，形如：https://svn1.intra.sina.com.cn/weibo/ria/t6/apps/enp\_mindPage /quickBuildTag/release\_tags/

**附录：**

有weibo ria权限的同学可以参考工程

https://svn1.intra.sina.com.cn/weibo/ria/t6/apps/enp\_mindPage/trunk