**Steel框架使用规范**

**一、steel的定义：**

    它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代码使用规范：**

**2.1 后端：**

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');

**2.2 前端开发约定：**

**2.2.1 目录结构：**(按字母顺序；gulp目录结构见weibo\_sell工程)

         |api    /\*假接口数据目录，格式见api目录下的README.MD\*/

            |build    /\*上线目录，用于推静态池，grunt生成\*/

            |debug    /\*调试目录，有grunt生成\*/

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

            |src    /\*源码目录\*/

                    |css    /\*样式源码\*/

                    |html    /\*demo目录\*/

                    |js    /\*逻辑源码\*/

                            |controller    /\*控制器目录\*/

                            |lib    /\*类库目录\*/

                                    |lib.js    /\*第三方库\*/

                            |logic    /\*逻辑源码\*/

                            |tpl    /\*模板存放目录\*/

                            |app.js    /\*应用入口文件，配置资源、路由\*/

            |SteelToolBox    /\*自动化工具集\*/

            |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({

                    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\*/

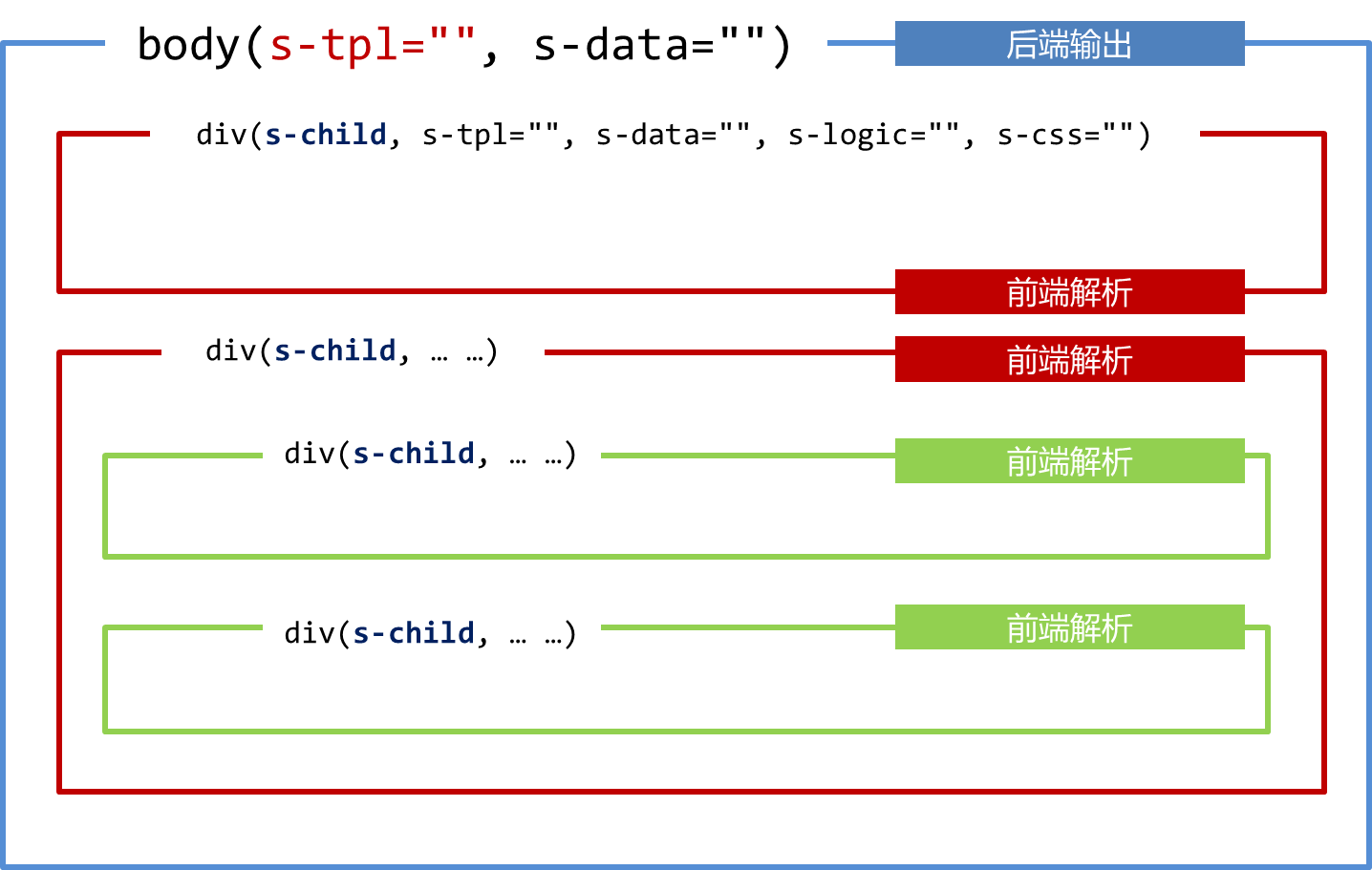
**2.2.4 模板开发约定：**

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（常用第二种方式）**

**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（常用）

【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

**2.3 前端安装、调试、上线**

**-------------------------- gulp 版本 start -----------------------------**

**2.3.1 环境说明**

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grunt build命令

QB上线

**-------------------------- gulp 版本 end -----------------------------**

**-------------------------- grunt 版本 start -----------------------------**

**2.3.1 环境说明**

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、gruntjs

**2.3.2 安装**

Nodejs ：请参考http://nodejs.org/ v0.10+

Gruntjs ：npm install -g grunt-cli

nodejs deps ：npm install

**2.3.3 使用服务**

默认端口80

可使用的命令

\*grunt serverDebug\*         ：启动调试服务器命令

\*grunt serverDist\*          ：启动仿真服务器命令

\*grunt serverStop\*          ：关闭服务器命令

//由于grunt-forever在windows下存在一个未知错误，用windows进行开发时请使用下面两条命令

\*grunt blockServerDebug\* ：启动调试服务器命令(用于windows)

\*grunt blockServerDist\*  ：启动仿真服务器命令(用于windows)

**2.3.4 使用处理工具**

单独使用Grunt命令中提供的处理命令可以了解处理中出现的错误。

可使用的命令如下：

\*grunt debug\*       ：对src目录文件进行debug处理

包括模板处理、脚本wrap和合并、静态文件copy等

\*grunt dist\*        ：对src目录文件进行dist处理

除做debug中的处理外，还有css压缩合并,js压缩混淆等

\*grunt build\*       ：用于上线文件的生成,会把可上线的结果处理了build目录下

**2.3.5 上线**

上线前使用grunt build命令

QB上线

**-------------------------- grunt 版本 end -----------------------------**