2016年6月24日 星期五

headmaster 三种打包方式

1.Combine v4老版本的代码。一般需要将conf目录往上提一级。
2.Voltron　现在线上用的最多的打包方式。去掉conf目录。（$Import加载模块）
3.Macross 类gulp的打包方式。通过moduel.export 、require方式做模块封装、模板封装。
4.deploy （发布js的目录。通过 sh /deploy/linux_start.sh 启动部署目录。可以通过界面部署js代码、添加项目.config/config.js 指定发布代码的路径。）
5.scorponok（js调试目录，node server.js 起服务（提供http静态资源服务。在scorponok 下的_config.json 配置调试环境的配置信息。））

```
{
//js部署代码的路径
        "DOCUMENT_ROOT" : “/Users/xiongguoqing/mypc/work",
//macross 调试的代码路径
        "MACROSS_ROOT"  : [“/t5/album"],
//voltron 调试的代码路径
        "VOLTRON_ROOT"  : [“/t5/taobao","/t6/home","/t6/page","/t5/brandevent","/mobile/apps/welfare"],
//combine 调试的代码路径
        "COMBINE_ROOT"    : ["/t5/home", “/t5/page","/t4/topic"],
//是否启用代码。填写代理的ip地址。多jser开发部署（其实只要部署在一台机器上就好）
        "PROXY_HOST"    : false,
//杀掉服务的key
        "CLEAR_KEY"     : “/__clear__",
//以下是各种加密方式的配置信息。详细参数名字如下：
        "FORMAT_CONF"   : {
                "MACROSS" : {
                        "is_beautify"    : true,
                        "is_mangle"              : false,
                        "is_squeeze"   : false,
                        "is_filename"   : true,
                        "is_debug"               : true
                },
                "VOLTRON" : {
                        "is_beautify"    : true,
                        "is_mangle"              : false,
                        "is_squeeze"   : false
                },
                "COMBINE" : {
                        "is_beautify"    : true,
                        "is_mangle"              : false,
                        "is_squeeze"   : false,
                        "is_confSpecial": true
```

```
            }
        }
}
```

通过在js目录中_config.json 来配置项目的压缩混淆模式。。，没有则走默认。

打包参数说明：
(1)mangle : 混淆
(2)squeeze : 压缩
(3)onlyconf : 只打包conf文件夹下的内容
(4)beautify : 保留代码的原格式（便于debug）
(5)confspecial : conf文件夹中的内容会向上提一个路径

前端代码合并打包工具（Combine

介绍

线上版(nodejs)前端合并打包工具，具有 将带有$Import的代码文件进行合并、变量混淆、代码压缩 等功能

约定

- 使用$Import(NS)做为代码依赖声明，其中NS为JS目录后的脚本命名空间；
- 代码依赖声明实例 $Import('ui.mod.layer');
- 处理输入目录下所有js目录里的代码

使用步骤

一、将工具代码下载到本地

```
1  cd /root/
2  svn co https://svn1.intra.sina.com.cn/weibo/ria/headmaster/combine
```

二、准备输入输出目录

```
1  mkdirhier /data1/wwwroot/js.wcdn.cn/t5/home/
2  cd /data1/wwwroot/js.wcdn.cn/t5/home/
3  svn co t5_home svn....
4  mkdirhier /data1/wwwroot/js.wcdn.cn/to/t5/home/
```

三、调用工具

```
node /root/combine/main.js /data1/wwwroot/js.wcdn.cn/t5/home/ /data1/wwwroot/js.wcdn.cn/to/t5/home/  –mangle –squeeze
```

命令行实例

- node /root/combine/main.js /data1/wwwroot/js.wcdn.cn/t5/home/ /data1/wwwroot/js.wcdn.cn/to/t5/home/ –mangle –squeeze

参数说明

- 第一个参数为要处理代码目录
- 第二个参数为要处理后代码的放置目录
- –beautify 是否格式化代码 默认为否
- –mangle 是否混淆变量 默认为否
- –squeeze 是否进一步压缩js 默认为否
- –onlyConf 是否只合并'/conf/'目录下js文件.默认合并所有文件 默认为否
- –confSpecial 是否把'/conf/'目录下文件提前输出到到'/conf/'的父目录. 如home/conf/base.js 合并后输出为home/base.js 默认为否

前端代码合并打包工具（Voltron）

介绍

脚本带版本号版(nodejs)前端合并打包工具，具有 将带有$Import的代码文件进行合并、变量混淆、代码压缩、生成基于内容的版本编号 等功能

约定

- 使用$Import(NS)做为代码依赖声明，其中NS为JS目录后的脚本命名空间；
- 代码依赖声明实例 $Import('ui.mod.layer');
- 处理输入目录下所有js目录里的代码，每个js目录做为一个处理单元
- _config.json放在js目录下 如： ... js.wcdn.cn/t5/home/js/_config.json
- 版本号对照文件输出到指定输出目录的相应js目录下 如: ... js.wcdn.cn/to/t5/home/js/md5_mapping_1366284188458.ini和 ... js.wcdn.cn/to/t5/home/js/md5_mapping_1366284188458.json

使用步骤

一、将工具代码下载到本地

```
1    cd /root/
2    svn co https://svn1.intra.sina.com.cn/weibo/ria/headmaster/voltron
```

二、准备输入输出目标

```
1    mkdirhier /data1/wwwroot/js.wcdn.cn/t5/home/
2    cd /data1/wwwroot/js.wcdn.cn/t5/home/
3    svn co t5_home svn....
4    mkdirhier /data1/wwwroot/js.wcdn.cn/to/t5/home/
```

三、调用工具

```
 node /root/voltron/main.js /data1/wwwroot/js.wcdn.cn/t5/home/ /
data1/wwwroot/js.wcdn.cn/to/t5/home/  –mangle –squeeze –md5
```

命令行实例

- node /root/voltron/main.js /data1/wwwroot/js.wcdn.cn/t5/home/ /
  data1/wwwroot/js.wcdn.cn/to/t5/home/ –mangle –squeeze –md5

参数说明

- 第一个参数为要处理代码目录
- 第二个参数为要处理后代码的放置目录
- –beautify 是否格式化代码 默认为否
- –mangle 是否混淆变量 默认为否
- –squeeze 是否进一步压缩js 默认为否
- –onlyConf 是否只合并'/conf/'目录下js文件.默认合并所有文件 默认为否
- –confSpecial 是否把'/conf/'目录下文件提前输出到到'/conf/'的父目录. 如
  home/conf/base.js 合并后输出为home/base.js 默认为否
- –md5 是否进行多版本脚本文件的生成

配置文件

说明：该工具支持工程下编写配置文件来替换命令参数输入，用以方便工具使用时的参数配置

- 约定该文件的文件名必须为_config.json
- 约定_config.json必须放在js目录下 如： ... js.wcdn.cn/t5/home/js/
  _config.json
- 文件配置参数说明

```
{
    "–beautify"        : false,//是否格式化代码
```

```
        "-mangle"         : true,//是否混淆变量
        "-squeeze"        : false,//是否进一步压缩js
        "-onlyconf"       : false,//是否只合并'/conf/'目录下js文件.默认合并所有文件
        "-confspecial"    : false,//是否把'/conf/'目录下文件提前输出到到'/conf/'的父目录.
                            //如home/conf/base.js 合并后输出为home/base.js
        "-filter"      : "pl/|conf/",//文件过滤正则表达式
                            //如设置了该值，工具只处理正则匹配成功的文件（匹配时使用文件路径中/js/后的部分时行判断）
        "-md5"            : true//是否进行多版本脚本文件的生成
}
```

前端代码组织构建工具（Macross

介绍

脚本带版本号版(nodejs)前端代码组织构建工具，通过require函数来遍历文件的依赖进行代码的重新组织、变量混淆、代码压缩、生成基于内容的版本编号 等功能

约定

- 工具认为代码包的跟路径时js目录，也就是说，一个js目录就是一个代码包。
- 使用var obj = require(NS);的方式将依赖引入并付值，其中NS是文件路径，其取值可以是相对路径，也可以以包的路径为根的绝对路径。
- 代码依赖声明实例 var alert = require('/common/dialog/alert'); 或者 var partOne = require('./parts/one');
- 代码中还有include方法，与require类似，但打包时不会作为依赖引入。
- 代码会以对module.exports付值或者对exports对象添加元素的方式提交返回值，其形式类似nodejs,可以参考。
- 处理输入目录下所有js目录里的代码，每个js目录做为一个处理单元
- _config.json放在js目录下 如： ... js.wcdn.cn/t5/home/js/_config.json
- 处理后的代码，一个文件会是一个function包裹的环境，其内部的变量外界完全无法访问。

- 版本号对照文件输出到指定输出目录的相应js目录下 如: ... js.wcdn.cn/to/t5/home/js/md5_mapping_1366284188458.ini和 ... js.wcdn.cn/to/t5/home/js/md5_mapping_1366284188458.json

使用步骤

一、将工具代码下载到本地

```
1  cd /root/
2  svn co https://svn1.intra.sina.com.cn/weibo/ria/headmaster/macross
```

二、准备输入输出目标

```
1  mkdirhier /data1/wwwroot/js.wcdn.cn/t5/home/
2  cd /data1/wwwroot/js.wcdn.cn/t5/home/
3  svn co t5_home svn....
4  mkdirhier /data1/wwwroot/js.wcdn.cn/to/t5/home/
```

三、调用工具

- node /root/macross/main.js /data1/wwwroot/js.wcdn.cn/t5/home/ /data1/wwwroot/js.wcdn.cn/to/t5/home/

命令行实例

- node /root/macross/main.js /data1/wwwroot/js.wcdn.cn/t5/home/ /data1/wwwroot/js.wcdn.cn/to/t5/home/

参数说明

- 第一个参数为要处理代码目录
- 第二个参数为要处理后代码的放置目录

配置文件

说明：该工具支持工程下编写配置文件来替换命令参数输入，用以方便工具使用时的参数配置

- 约定该文件的文件名必须为_config.json
- 约定_config.json必须放在js目录下 如： ... js.wcdn.cn/t5/home/js/_config.json
- 文件配置参数说明

```
{
    "is_beautify":false,//是否格式化代码
    "is_mangle" : false,//是否混淆变量
```

```
        "is_squeeze" : false,//是否进一步压缩js
        "is_file_version" : false,//是否进行多版本脚本文件的生成
        "is_debug" : false,//是否打包为debug模式
        "is_filename" : false,//是否使用原有文件名不进行混淆
        "is_filter" : false,//是否开启文件过滤
        "filter" : '' //文件过滤的规则（正则表达式），如"pl/|conf/"
}
```

headmaster部署环境相关文档：
一.使用场景

（1）方便测试部的同事独立部署前端JS和CSS代码。减轻前端工程师的负担。 让前端工程师有更多的时间专注于程序开发，保证代码质量。

（2）使用此项目部署后，在前端代码上，可以完全模拟线上环境，保证代码上线质量。

（3）对于打包流程不是特别了解的同学，可以在开发阶段，进行部署测试，尽早发现问题，节省开发成本，而不是等到仿真测试的时候才发现由于文件打包造成的开发代码与上线代码效果不一致的行为。

二.部署达到的效果

（1） 可以针对JS和CSS 的项目，使用svn url
https://svn1.intra.sina.com.cn/weibo/ria/t5/home/branches/20121219_giftShengdan_MINIBLOGREQ17688/
和产品线名称(V5主站JS) 进行相应项目的部署。

（2） 只针对要进行测试的项目进行部署，其他不进行测试的项目使用反向代理，读取线上的文件，保证测试环境不会出现文件找不到或与线上不一致的行为。

（3） 提供已经部署的项目列表，方便追踪，多个测试工程师可以使用同一个机器部署多个项目。

三.部署截图说明

（1）从部署页面点击“添加产品线”，进入到“部署项目配置页面”。填写产品线名称，部署路径，

产品类型（JS或CSS），JS的项目要添加打包参数，填写完成后，该配置信息就会保存下来，一个产品线

只需要添加一次即可。
Username,Password,SVN URL,Product Name

（2）填入svn用户名，svn密码，svn url ，选择相应的产品线，单击部署，即可部署一个独立的项目
Username,Password,SVN URL,Product Name

（3）在提交的页面上显示部署进度，直至部署成功。
cmdLine

（4）在部署页面有列表显示部署的项目，时间，产品线名，部署人，提供取消部署操作
deployList

（5）部署完成后，设置HOST即可测试，可以从X-Powered-By响应头知道，文件是部署的机器上的还是线上的
fromLocal

本地部署的文件

X-Powered-By 的值为 weibo,source=local
fromCdn

反向代理线上的文件 X-powered-By的值为 weibo,source=cdn

四．部署方法(支持windows，linux，mac)

（1） 安装nodejs环境 http://nodejs.org/download/

（2） 能够使用shell 或cmd命令操作svn

Windows：可以安装CollabNet Subversion Command-Line Client (for Windows)

http://www.pblee.net/file.axd?file=2011%2f4%2fSetup-Subversion-1.6.16.msi

安装完成后，需要将svn.exe添加到环境变量中

Linux，Mac:

安装subversion

http://subversion.apache.org/packages.html

（3） 安装apache 或 nginx

Apache：推荐xampp， 傻瓜化安装，快速方便 http://www.apachefriends.org/zh_cn/xampp.html

Nginx: http://nginx.org/en/download.html

（4） 检出部署使用的nodejs代码到测试机本地文件夹

cd /data1/wwwroot/ svn co https://svn1.intra.sina.com.cn/weibo/ria/sandbox/lianyi/deploytesting ./deploy

（5） 修改配置文件

修改config/config.js documentRoot指向apache或nginx 的 vhost的根目录 svnRoot 指向微博ria的svn地址

修改config/product.txt（此步骤一般为前端工程师完成） product对应产品线名称 path对应检出的文件夹 type表示是js还是css packargs 对应打包程序中的选项（具体项目具体填写）

mangle	混淆
squeeze	压缩
confspecial	针对conf文件夹中的内容做提升目录操作
onlyconf	只上线conf文件夹中的内容

（6） 为apache或nginx针对ria目录添加vhost配置，添加反向代理功能。（需要保证对应的vhost目录存在，对应的日志目录存在） Apache: 对应我的机器为(/opt/lampp/etc/extra/httpd-vhosts.conf)
Apache Vhost

```
<VirtualHost *:80>
ServerAdmin lianyi@staff.sina.com.cn
DocumentRoot "/data1/wwwroot/js.wcdn.cn"
ServerName js.t.sinajs.cn
ServerAlias js1.t.sinajs.cn js2.t.sinajs.cn tjs.sjs.sinajs.cn
RewriteEngine On
RewriteCond %{DOCUMENT_ROOT}%{REQUEST_FILENAME} -f
RewriteRule ^/(.*)$ /$1 [NC,QSA,L,E=local:true]
Header unset X-Powered-By
Header set X-Powered-By weibo,source=local env=local
Header set X-Powered-By weibo,source=cdn env=!local
ProxyPass / http://js.t.sinajs.cn/
```

```
ProxyPassReverse / http://js.t.sinajs.cn/
ErrorLog "/data1/apachelogs/errorlog"
CustomLog "/data1/apachelogs/customlog" common
</VirtualHost>
```

Nginx：对应我的机器为（/usr/local/nginx/conf/nginx.conf）
NginxVhost

```
==========================================

  server {
    listen 80;
    server_name js.t.sinajs.cn js1.t.sinajs.cn js2.t.sinajs.cn
tjs.sjs.sinajs.cn;
    access_log /data1/wwwroot/logs/accesslognginx combined;
    location / {
      root /data1/wwwroot/js.wcdn.cn;
      if (-f $request_filename) {
          add_header X-Powered-By weibo,source=local;
          break;
      }
      if (!-f $request_filename) {
          add_header X-Powered-By weibo,source=cdn;
          proxy_pass http://js.t.sinajs.cn;
          break;
      }
    }
  }

================================================
```

（7） 启动nodejs脚本

Linux：

cd server
sh linux_start.sh

错误日志会输出到server/log.txt文件中

Windows：

双击 server目录下的win_start.bat运行。

（8） 在浏览器中访问，假设部署机器IP 10.210.210.65

部署JS

http:// 10.210.210.65:8765/deployjs

部署CSS

http:// 10.210.210.65:8765/deploycss

五．安装过程视频

(1) 具体使用方法 http://10.210.210.65/videos/configuration/configuration.html

(2) 安装依赖环境(nodejs、svn、apache、nginx) http://10.210.210.65/videos/softsetup/softsetup.html

快速一条命令安装的svn路径。

https://svn1.intra.sina.com.cn/weibo/ria/sandbox/zhouliang2/riaServerEnv/

H5 gulp 使用

1. linux在进入 部署/js的目录。
2. 首次执行 npm intall命令，下载依赖的node包文件。
3. 执行 gulp 命令就可以。（每次修改后都会自动打包。请求为dest的文件。提交的时候源代码和dest目录的文件都提交。）