**打包发布工具**

# 概述

为了优化项目前端的代码和提高发布效率，本文档定义了一套打包策略供选择使用，按照文档的规范进行开发的项目可以使用框架提供的打包工具进行一键发布。

本文档主要说明打包的原理、项目开发过程中代码及文件规范、项目开发需要注意的事项等内容。

打包工具采用html注释来标记打包指令，从而降低对原有系统的侵入性。

打包工具支持采用NEJ依赖系统的项目并分离NEJ依赖系统，支持NEJ平台适配策略，并按平台输出内容，同时也支持非NEJ项目通过增加打包指令来输出。

# 工具使用

## 环境配置

发布工具基于[nodejs](http://nodejs.org/)平台，因此需要使用者先安装[nodejs](http://nodejs.org/)环境，nodejs在各平台下的安装配置请参阅官方说明。<http://nodejs.org/>

## 获取工具

NEJ的发布工具可以从以下git地址（<https://github.com/genify/publish.git>）获取，也可以使用npm install nej-pub -g 命令安装，多个项目打包发布只需获取一份工具即可。

## 使用说明

1. 执行npm install nej-pub –g命令安装打包工具，如果已安装打包工具可忽略此步骤
2. 执行nej-init命令初始化打包配置文件，命令后面可输入配置文件输出路径，默认输出在当前目录，此时会在指定目录或者当前目录生成一个release.conf文件用来配置打包选项
3. 按照项目的实际情况修改配置文件release.conf，具体参数见[配置参数](#_配置参数)章节
4. 执行nej-pub release.conf命令打包项目，如果release.conf文件在当前目录也可不指定

# 打包策略

## 工具配置

项目在使用打包工具时需要配置一些参数，包括路径，优化策略等等，详情见[工具使用](#_工具使用)章节

## 打包样式

打包工具通过分析所有入口文件中的STYLE标记里的样式文件内容进行如下处理

* 所有样式文件做压缩处理
* 在一个以上入口页面文件中使用到的样式文件合并至一个文件（该文件名可配置）
* 以合并后的文件内容计算出MD5值作为文件的版本号
* 入口页面特有的样式内容以内联的方式加入页面中
* 以上生成的代码替换页面中STYLE标记下的代码



## 打包模板

模板的打包关注点在外联资源的处理，对于类型为css和html的外联资源打包时直接将该类资源压缩后内联至文件中，脚本的打包策略见打包脚本的说明

## 打包脚本

打包工具通过分析入口文件中的ENTRY标记下的入口脚本及模板文件中类型为js的模板进行如下处理

* 所有脚本文件做混淆压缩处理
* 过滤DEFINE标记定义的依赖系统脚本引入
* 收集每个页面可能用到的所有脚本，将在一个以上的html文件中出现过的脚本合并到一个文件（该文件名可配置）
* 以合并后的文件内容计算出MD5值作为文件的版本号
* 入口页面及模块模板中特有的脚本以内联的方式加入页面中
* 以上生成的代码替换页面中相应位置的代码



## 打包输出

打包输出的结构中html和template目录的结构同输入目录保持一致，只是调整了内容中的资源，输出的静态文件由配置文件确定具体输出内容

# 结构建议

推荐的项目前端目录结构如下图所示



* deploy：用来放置打包工具的配置文件及执行文件
* template：如果项目用的是服务器端模板，则使用该目录放置模板文件

Web访问入口下放置内容

* res：用来放置项目相关的资源文件，如图片、Flash、视频等
* css：用来放置项目相关的样式文件
* html：用来放置项目结构相关的文件，系统的入口文件也在此目录下
* javascript：用来放置项目业务逻辑相关的脚本文件

## html

项目的入口文件及页面组成模块的模板文件均在此目录下，对于文件的命名没有特殊要求，页面入口文件直接位于html目录下，对于单页面系统的文件结构类似如下图所示



打包工具进行打包时从此目录作为入口处理目录下的所有html文件，输出时保持此目录下的文件结构，文件里的内容会做相应的压缩处理

## template

如果项目使用服务器端模板，则页面模板文件放于此目录下，模块入口文件仍在html目录下，范例结构如下所示



## css

项目的公用样式文件均放在此目录下，这里对于文件命名没有特别要求，此目录结构没有特殊要求，打包工具进行打包时在此目录下仅输出压缩后的入口文件的样式，模块的样式超过指定大小均以内联的方式合并至模块结构的模板中

## javascript

项目相关的业务逻辑脚本均放在此目录下，这里对于文件命名没有特别要求，此目录结构没有特殊要求，打包工具进行打包时输出入口和模块的压缩混淆脚本文件

## res

项目中使用的所有资源文件均放在此目录下

## deploy

发布工具在项目中的配置文件及执行文件所在的位置，关于此目录见后面[工具的使用](#_项目配置)章节的介绍

# 打包标记

项目页面的入口文件及模块的模板文件中可以通过指定的标记对内容进行打包配置，打包标记基于html的注释标记扩展，因此不会影响到页面的实际呈现

## 标记格式

打包工具支持的标记格式如下所示



* 标记以“<!--”开始，以”-->”结束，一个标记在一行内完成，一行仅允许包含一个标记内容或空格
* 标记为大写单词，并以“@”为前缀，如”@STYLE”
* 成对出现的标记必须使用结束标记，结束标记为开始标记的“@”符号前加“/”，如”/@TEMPLATE”作为“@TEMPLATE”的结束标记
* 标记支持的配置参数为JS对象，紧跟标记名称后，并以空格分隔，具体标记支持的参数见各标记说明，范例如



* 所有打包标记在项目打包完成后删除，因此在最终输出的html文件中不会有任何标记

## NOCOMPRESS

是否需要结束：可选

支持配置参数：无

此标记用以表明从当前位置开始到标记结束位置之间的html代码打包时无需压缩，如没有结束标记则此效果延续到文件结束位置，嵌套情况下此标记仅允许嵌套在[TEMPLATE](#_TEMPLATE)标记中，具体使用范例如下所示：

* 全文无需压缩在文件起始位置加标记



* 部分内容压缩需指定不压缩的内容



## STYLE

是否需要结束：无

支持配置参数：core [Boolean] core.css文件插入形式

true 强行插入core.css

false 禁止插入core.css

不设置 自动解析【默认】

inline [Boolean] 是否强行内联core样式，默认外联

此标记用以表明当前位置插入打包后的样式文件，具体使用范例如下所示：



项目自己定义了core文件列表，并通过服务器端模版引入各个页面时，可以增加core参数配置

## TEMPLATE

是否需要结束：必须

支持配置参数：无

此标记用以表明从当前位置至结束标记之前的内容中存在资源模板信息，打包针对模板会做优化处理包括：

* 外联样式文件（类型为css的模板）做内联处理
* 外联嵌套模板文件（类型为html的模板）做内联处理
* 外联脚本文件（类型为js的模板）按照指定标记做内联或者外联处理

因此仅需要在引入资源模板是加此标记即可，具体使用范例如下所示：



## MODULE

是否需要结束：必须

支持配置参数：无

此标记用以表明从当前位置至结束标记之前的内容中存在模块资源，最终打包时一个模块资源会形成一个模版集合，在模块调度时直接动态解析这些模版而不会发请求加载相关资源

注：此标记内的内容为外联的html类型模版，且符合模块调度器方案中的模块规范

此标记必须出现在[DEFINE](#_DEFINE)标记所在的页面，否则将被忽略



## VERSION

是否需要结束：无

支持配置参数：无

此标记用以表明当前位置插入模块版本信息，后续脚本内仅允许出现模块的版本配置信息，否则打包后内容将丢失，如果项目没有使用NEJ的模块调度模型则可以忽略此标记

具体使用范例如下图所示：



## NOPARSE

是否需要结束：必须

支持配置参数：无

此标记用以表明从当前位置至结束标记之前的内容不需要做任何处理

具体使用范例如下图所示：



## IGNORE

是否需要结束：必须

支持配置参数：mode [String] 忽略模式

test 测试版本

online 上线版本，默认值

develop 开发版本

此标记用以表明从当前位置至结束标记之前的内容在打完包后删除，mode配置忽略的条件，多个值使用“|”分隔，当打包工具中的[X\_RELEASE\_MODE](#_X_RELEASE_MODE)配置值出现在mode的值中时即忽略此标记间的内容

具体使用范例如下图所示：







## DEFINE

是否需要结束：无

支持配置参数：nodep [Boolean] 是否没有使用依赖系统

true 没有使用依赖系统，后续脚本为源码

false 使用依赖系统，后续脚本为define.js

core [Boolean] core.js文件插入形式

true 强行插入core.js

false 禁止插入core.js

不设置 自动解析【默认】

inline [Boolean] 是否强行内联core脚本，默认外联

此标记用以表明后续的脚本标签为依赖系统定义文件路径，对于该文件打包工作做以下处理：

* 最终输出文件不再引入此文件
* 从该脚本路径解析出框架所在路径
* define.js路径支持平台参数配置，所带参数详细说明见[平台参数](#_平台参数)章节

使用依赖系统，后续第一个外联脚本为define.js：



没有使用依赖系统，后续脚本作为源码，此标记仅用来表示打包后脚本插入位置：



项目自己定义了core文件列表，并使用服务器端模版引入页面时可以使用core参数配置；当core显式的设置为false时当前文件的脚本将独立解析，不参与core脚本的合并策略

## MANIFEST

是否需要结束：无

支持配置参数：无

此标记用以表面后续的<html>标签需要插入manifest文件

# 配置参数

## 配置说明

* 每行第一个非空字符为“#”则表示该行为注释
* 配置参数使用 KEY = VALUE 格式
* 配置参数为路径的，如果是相对路径则必须以“./”、“../”开始
* 配置参数中如无特殊说明使用的相对路径均相对于DIR\_WEBROOT的路径
* 配置参数名称忽略大小写，即cfg\_dir\_webroot等价于CFG\_DIR\_WEBROOT

## DIR\_WEBROOT

必须

WEB根路径，如果是相对路径则相对于当前配置文件路径(即release.conf文件所在目录)

## DIR\_SOURCE

可选

项目HTML文件根路径，如果是使用服务器端模板的项目可以不用配置此目录，直接配置[DIR\_SOURCE\_TP](#_DIR_SOURCE_TP)即可

## DIR\_OUTPUT

可选

打包HTML文件输出路径，默认为[DIR\_WEBROOT](#_DIR_WEBROOT)配置信息

## DIR\_OUTPUT\_STATIC

可选

静态文件输出目录，默认为[DIR\_OUTPUT](#_DIR_OUTPUT)配置信息，如果该配置目录不在[DIR\_WEBROOT](#_DIR_WEBROOT)配置的目录下，则自动调整为[DIR\_WEBROOT](#_DIR_WEBROOT)下以[DIR\_OUTPUT](#_DIR_OUTPUT)目录名命名的目录下，以确保所有静态资源对外可访问

## DIR\_SOURCE\_TP

可选

项目服务器端模板文件根路径，服务器端模板文件确保页面所需的CSS、JS文件的引用均出现在模板文件中

## DIR\_OUTPUT\_TP

可选

项目服务器端模板输出路径，默认为[DIR\_OUTPUT](#_DIR_OUTPUT)配置信息，如果没有配置[DIR\_SOURCE\_TP](#_DIR_SOURCE_TP)则忽略此参数配置信息

## DIR\_STATIC

可选（必须确保路径存在）

静态资源路径，默认为[DIR\_WEBROOT](#_DIR_WEBROOT)下的res目录，如果静态资源的目录不是[DIR\_WEBROOT](#_DIR_WEBROOT)下的res且在html代码中引用了静态资源地址则需要配置此参数来调整html中静态资源的引用路径

## NEJ\_DIR

可选

NEJ框架本地路径，此配置优先级高于自动识别

## NEJ\_PLATFORM

可选

NEJ平台适配参数，等价于define.js?p=wk|td这里p参数的配置，优先级比p参数高，配置规则同p参数

## MANIFEST\_OUTPUT

可选

HTML5离线应用配置文件输出路径，对于需要使用HTML5离线的应用可配置此文件路径输出manifest文件

## MANIFEST\_TEMPLATE

可选

HTML5离线应用配置文件输出模板路径，纯文本文件，不配置使用默认模板；如果是相对路径则相对于当前配置文件路径(即.conf文件所在目录)；模板中使用的变量格式为 #<VAR\_NAME>，其中VAR\_NAME为变量名，可用变量名如下：

VERSION - 版本信息

CACHE\_LIST - 缓存资源列表，换行符分隔

## MANIFEST\_FILTER

可选

HTML5离线应用配置文件输出输出CACHE\_LIST列表过滤正则表达式，确保可以通过new RegExp转换成正则

## DM\_STATIC

可选

静态资源域名配置

* 如果没有配置，则项目的静态资源中相对路径的请求相对于页面路径
* 如果配置了"/"，则项目的静态资源中相对路径的请求相对于WEBROOT的路径
* 如果配置了域名，则项目的静态资源中相对路径的请求使用当前域名的绝对路径

默认静态资源请求域名，限定[DIR\_STATIC](#_DIR_STATIC)配置路径下资源、JS资源、CSS资源访问域

## DM\_STATIC\_CS

可选

外联样式请求域名，没有配置则使用[DM\_STATIC](#_DM_STATIC)配置信息，规则同[DM\_STATIC](#_DM_STATIC)

## DM\_STATIC\_JS

可选

外联脚本请求域名，没有配置则使用[DM\_STATIC](#_DM_STATIC)配置信息，规则同[DM\_STATIC](#_DM_STATIC)

## DM\_STATIC\_MR

可选

模块根路径配置，模块调度方案中模块文件所在的根路径，默认自动解析

## DM\_STATIC\_MF

可选

MANIFEST文件请求域名，没有配置则使用[DM\_STATIC](#_DM_STATIC_1)配置信息

## OBF\_LEVEL

可选

脚本混淆等级

* 0 - 不做混淆，只做压缩
* 1 - 只混淆单个下划线(\_)前缀的变量，如\_xxx
* 2 - 在1的基础上增加两个下划线(\_\_)前缀的变量，如\_xxx、\_\_xxx
* 3 - 所有下滑线前缀的变量都做混淆【默认配置】，如\_xxx、\_\_xxx、\_$xxx、\_$$xxx

## OBF\_LINE\_MODE

可选

压缩分行模式

* 0 - 所有文件合并成一行
* 1 - 一个文件一行【默认配置】

## OBF\_NAME\_BAGS

可选

脚本混淆变量名称对照表文件路径，如果是相对路径则相对于当前配置文件路径，默认为release.conf文件同目录下的name.txt文件

## OBF\_MAX\_CS\_INLINE\_SIZE

可选

内联样式内容的最大长度（单位K），默认为50K，如超出该长度则将样式做为外联文件导入

## OBF\_MAX\_JS\_INLINE\_SIZE

可选

内联脚本内容的最大长度（单位K），默认为0K，即使用外联文件导入

## CORE\_LIST\_JS

可选

文件合并策略配置，core文件包含的脚本列表，配置的列表会自动做依赖分析

* 配置路径如果是相对路径则相对于当前配置文件路径
* 如果没有配置core文件列表则一个文件在2个以上（包含2个）文件中出现就会合并到core文件中

javascript core文件列表配置文件

* 如果指定路径则表示配置文件地址，如./core.js.txt
* 如果指定列表则表示文件列表，必须在一行内完成，如["{lib}util/ajax/xdr.js",...]

## CORE\_LIST\_CS

可选

css core文件列表配置文件，规则同[CORE\_LIST\_JS](#_CORE_LIST_JS)

* 如果指定路径则表示配置文件地址，如./core.css.txt
* 如果指定列表则表示文件列表，必须在一行内完成，如["/src/css/reset.css",...]

## CORE\_MASK\_JS

可选

core文件屏蔽的脚本列表，在core文件解析完成后根据此配置列表删除core中出现的文件，注意：

* 该配置列表不做文件依赖分析
* 该配置列表中配置的文件不应该被其他文件依赖
* 配置路径如果是相对路径则相对于当前配置文件路径

javascript core文件屏蔽列表配置同[CORE\_LIST\_JS](#_CORE_LIST_JS)

* 如果指定路径则表示配置文件地址，如./mask.js.txt
* 如果指定列表则表示文件列表，必须在一行内完成，如["{lib}util/ajax/xdr.js",...]

## CORE\_MASK\_CS

可选

core文件屏蔽的样式列表，在core文件解析完成后根据此配置列表删除core中出现的文件，配置路径如果是相对路径则相对于当前配置文件路径

css core文件屏蔽列表配置同[CORE\_LIST\_CS](#_CORE_LIST_CS)

* 如果指定路径则表示配置文件地址，如./mask.css.txt
* 如果指定列表则表示文件列表，必须在一行内完成，如["{lib}util/ajax/xdr.js",...]

## ALIAS\_START\_TAG

可选

别名开始标记，默认为 ${

路径别名配置

页面引入的脚本或者样式可以使用服务器端模板标记，如

<script src="${config\_lib\_root}define.js"></script>

## ALIAS\_END\_TAG

可选

别名结束标记，默认为 }

## ALIAS\_DICTIONARY

可选

别名配置列表，如

{"nej":"D:/nej/src","config\_lib\_root":"{lib}"}

## NAME\_SUFFIX

可选

输出文件名后缀，配置了后缀的情况下对所有输出文件的请求都不会带版本号，默认带版本号

## FILE\_SUFFIX

可选

输入文件后缀匹配规则（主要是[DIR\_SOURCE](#_DIR_SOURCE)和[DIR\_SOURCE\_TP](#_DIR_SOURCE_TP)配置的输入目录下的文件），多个后缀用“|”分隔，忽略大小写，默认分析指定目录下的所有文件，复杂过滤器可使用[FILE\_FILTER](#_FILE_FILTER)正则匹配过滤

## FILE\_FILTER

可选

输入文件路径匹配规则（主要是[DIR\_SOURCE](#_DIR_SOURCE)和[DIR\_SOURCE\_TP](#_DIR_SOURCE_TP)配置的输入目录下的文件），满足以下要求：

* 正则表达式，忽略大小写
* 确保可以通过new RegExp转换成正则
* 规则用来检验要处理的文件，默认处理所有文件

## FILE\_CHARSET

可选

输入输出文件编码，默认为utf-8

注意：项目必须保证所有文件的编码一致，如css/js/ftl等文件的编码一致

## RAND\_VERSION

可选

输出文件版本号使用随机算法

* 默认输出文件版本号根据内容计算，如果文件内容不变版本不变
* 此参数设置为true时每次打包生成的版本号都不一样，常用于线上服务器更新失败时强制变更版本信息

## STATIC\_VERSION

可选

输出文件中的静态资源地址【[DIR\_STATIC](#_DIR_STATIC)配置目录下的资源文件】是否自动带上版本信息，默认情况下不自动带上版本信息。另外以下几种情况的资源也不会自动带上版本信息：

* 资源地址指定了版本信息，如/res/logo.png?v=1
* 资源地址非本地文件，如http://a.b.com/logo.png

## X\_NOCOMPRESS

可选

输出文件不做压缩，效果等价于在页面头部增加[NOCOMPRESS](#_NOCOMPRESS)标记，默认根据页面标记处理，此优先级高于页面标记[NOCOMPRESS](#_NOCOMPRESS)

## X\_NOPARSE\_FLAG

可选

输出文件中不做解析的内容设置，效果等价于设置[NOPARSE](#_NOPARSE)标记，默认根据页面标记处理，此优先级高于页面配置标记[NOPARSE](#_NOPARSE)，允许配置以下值：

* 0 - 根据页面标记处理
* 1 - 不处理内联样式，如<style></style>之间的内容
* 2 - 不处理内联脚本，如<script></script>之间的内容
* 3 - 不处理内联样式和脚本，即1和2的情况都不处理

## X\_NOCORE\_STYLE

可选

每个页面样式单独分析处理，不再根据合并策略生成core文件，打开此开关将忽略[CORE\_LIST\_CS](#_CORE_LIST_CS)中的文件列表配置

## X\_NOCORE\_SCRIPT

可选

每个页面脚本单独分析处理，不再根据合并策略生成core文件，打开此开关将忽略[CORE\_LIST\_JS](#_CORE_LIST_JS)中的文件列表配置

## X\_RELEASE\_MODE

可选

发布模式，可以使用以下三种模式，主要用于控制[IGNORE](#_IGNORE)标记的处理，默认为online模式

* test 测试模式
* online 线上模式，默认模式
* develop 开发模式

## X\_AUTO\_EXLINK\_PATH

可选

发布后使用绝对路径调整外链地址，主要处理页面中同时符合以下条件的外链地址：

* 地址指向的内容为[DIR\_SOURCE](#_DIR_SOURCE)配置下的内容
* html中使用src="[LINK]"或者href="[LINK]"形式引入的地址

## X\_AUTO\_EXLINK\_PREFIX

可选

如果[X\_AUTO\_EXLINK\_PATH](#_X_AUTO_EXLINK_PATH)配置为true则可以通过此配置增加前缀标识来强行替换带该标识的路径，带标识路径规则为[X\_AUTO\_EXLINK\_PREFIX]="[LINK]"，多个前缀标识用|分隔，注：存在单页面多模块调度的系统这里切勿配置data-src

# 使用范例

范例中涉及的标记及配置以上文提到的为主

## 标记使用范例



## 配置文件使用范例



