Gulp构建工具的基本使用

> 本文主要介绍前端开发中常用的构建工具Gulp，具体包括Gulp的基本情况、安装、使用以及常见插件的安装、配置和使用等内容。

Gulp介绍

Gulp 是使用JavaScript编写的，运行在Node.js上的一款自动化构建工具 ，同类型的构建工具还有Grunt、Npm-script等。Gulp的构建系统基于流来实现，增加了监听文件、读写文件以及流式处理等功能，Gulp作为后起之秀整体来说比Grunt更加强大而且使用起来也更加简单，你值得拥有。

Gulp优点

> ❏ 基于流的操作，能更快速的构建项目并减少频繁的IO操作，更高效。
> ❏ 提供最少的API，降低开发者的学习成本，自动构建相关代码更简单。
> ❏ 代码优先的策略，让简单的任务简单处理，让复杂的任务变得可管理。
> ❏ 严格的插件指南，确保所有的插件简单，职责单一能按期望方式工作。

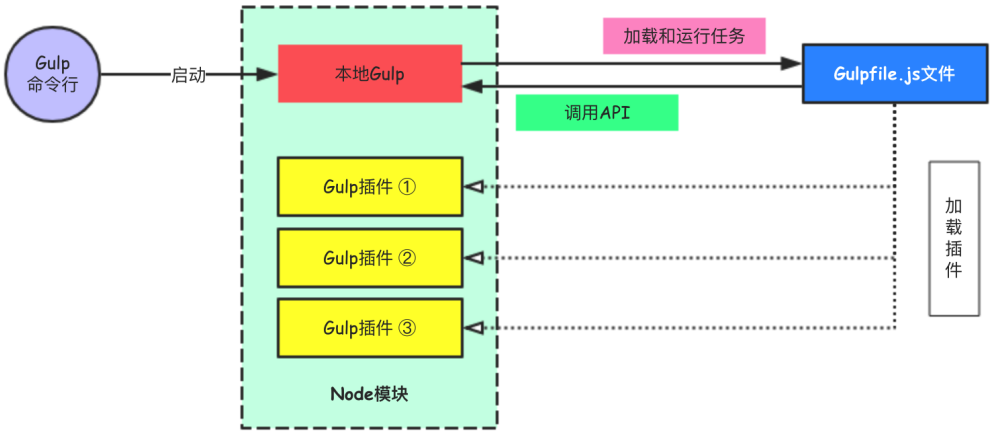
Gulp和Grunt简单对比

构建工具	Gulp	Grunt
工作流特点	基于流	基于文件
优先	代码优先	配置优先
插件数量	3556 +	6411 +

Gulp的组成结构

Gulp本身和Grunt还是很相像的，Gulp项目主要由 `Gulp-Cli` 、 `Node模块(本地的gulp和gulp插件)` 以及 `Gulpfile文件` 构成。

> **Gulp-Cli** ✧ 启动Gulp构建工具的命令行接口，全局安装。
> **本地Gulp** ✧ 构建时实际运行的本地程序,提供基本的API并加载构建指令和运行任务。
> **Gulp插件** ✧ Gulp构建工具生态系统中拥有众多高质量的插件，本质上是对特定任务的封装。
> **Gulpfile文件** ✧ 定义所有构建任务的配置文件，告诉本地Gulp如何执行具体构建操作的指令文件。



上图是Gulp几个部分的关系图，这里做简单说明。

`Gulp-cli` 是Gulp的命令行工具，如果项目中要使用Gulp构建工具，那么必须先安装Gulp-cli，它的作用非常简单，就死检查当前项目里是否安装了本地的Gulp，如果安装了那么Gulp-cli会根据命令行中输入的命令参数来启动本地Gulp接管后续的操作。Gulp-cli通常需要全局安装，这样任何目录下面的Gulp才能够在终端中执行。

`本地Gulp` 主要有两个作用，一个是提供构建必要的基本API，另一个是加载并执行定义好的任务。需要注意的是，本地的Gulp才是真正的Gulp运行时，它负责处理所有的任务，而Gulp-Cli是本地Gulp的全局入口，它负责把所有的命令参数都转交给本地Gulp处理，之后由Gulp来接手具体的构建过程。

本地Gulp在运行的时候，需要读取 `Gulpfile.js文件` 的内容，Gulpfile.js文件主要加载Gulp插件，并且调用Gulp相关的api来定义task（任务）。Gulp具体要做什么操作，应该如何执行这些操作等都由Gulpfile.js文件决定。

Gulp参考

Gulp官网
Gulp官网（中文）
Gulp官方插件列表页
Gulp在npm的发布页
Gulp中文文档-GitBook

Gulp的安装

环境支持

Gulp基于Node.js，所以在安装gulp之前，请确认已经在系统环境中安装了Node.js和npm（默认和Node绑定在一起安装）。

Node.js的安装请参考Node官网，根据提示安装完成后，可以通过下面的命令来检查Node.js和npm是否被正确安装。

```
$ node --version
$ npm --version
```

Gulp-Cli的安装

Node的环境准备好之后，接下来就可以来安装Gulp的命令行工具了。需要注意，Gulp的命令行工具需要全局安装，安装完成后可以通过查看版本号的方式来进行确认，下面给出安装命令。

```
$ npm install -g gulp-cli
$ gulp --version
```

这里顺便贴出终端执行细节。

```BASH
wendingding$ node --version
v8.9.3
wendingding$ npm --version
5.5.1
wendingding$ npm install -g gulp-cli
/usr/local/bin/gulp -> /usr/local/lib/node_modules/gulp-cli/bin/gulp.js
+ gulp-cli@2.0.1
added 236 packages in 89.58s
wendingding$ gulp --version
[17:23:12] CLI version 2.0.1
```

本地Gulp的安装

安装好Gulp的命令行工具后，就可以在您的项目目录中安装本地的局部Gulp了。本地的Gulp是真正运行构建任务的程序，全局的Gulp-Cli仅仅用于检查本地Gulp是否可用，如果可用那么就启动本地Gulp程序。安装成功后，本地的Gulp位于目录结构中的node_modules文件夹下。它包含了Gulpfile文件需要的所有函数和API。

① 创建项目目录，使用命令行工具先进入当前路径（假设项目名称为GulpDemo）

```
$ mkdir GulpDemo
$ cd GulpDemo/
```

② 初始化环境，创建package.json文件

```BASH
wendingding$ mkdir GulpDemo
wendingding$ cd GulpDemo/
GulpDemo wendingding$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
```

```
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (gulpdemo)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author: 文顶顶

license: (ISC)
About to write to /Users/文顶顶/GulpDemo/package.json:
{
  "name": "gulpdemo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "文顶顶",
  "license": "ISC"
}

Is this ok? (yes) yes
```

③ 安装Gulp到当前项目中

```
$ npm install --save-dev gulp
```

`save-dev` 参数的作用是把安装的Gulp版本正确保存在package.json文件中，命令行执行完毕后我们可以通过 `cat指令` 来查看package.json文件的内容，可以发现devDependencies字段增加了gulp信息。

BASH

```
GulpDemo wendingding$ cat package.json
{
  "name": "gulpdemo",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "文顶顶",
  "license": "ISC",
  "devDependencies": {
    "gulp": "^3.9.1"
  }
}
```

④ Gulp安装好后，可以通过 `--version` 命令检查版本信息确认

```bash
GulpDemo wendingding$ gulp --version
[19:30:11] CLI version 2.0.1
[19:30:11] Local version 3.9.1
GulpDemo wendingding$ Tree -L 1
.
├── node_modules
├── package-lock.json
└── package.json
```

现在全局的Gulp-Cli和本地的Gulp安装完毕，且package.json文件准备好后，就可以创建Gulpfile.js文件来处理具体的Task了。

Gulp的基本使用

前面我们对Gulp进行了简单了介绍，并且已经在全局安装好了Gulp命令行工具并把Gulp安装到了当前项目（GulpDemo）中，接下来这里简单介绍下Gulp的基本使用。

Gulp要执行哪些任务，这些任务的具体执行方式等等的都由Gulpfile文件中的命令来决定，现在我们先在GulpDemo根目录下创建一个空的Gulpfile.js文件，然后编写具体的Task处理代码。

在根目录创建文件命令： `$ touch Gulpfile.js`

Gulp的任务可以简单的划分为两种，一种是自定义任务，另外一种是使用Gulp插件，不管是什么类型的任务都通过gulp.task这个API来进行设置。

自定义Task

```
//[01] 引入本地的Gulp
var gulp = require("gulp");

//[02] 定义Task（自定义的任务）
gulp.task("demo",function(){
  console.log("这是一个自定义的Task");
})
```

编辑Gulpfile.js文件的内容，这里我们先引入了本地的gulp，然后通过gulp.task这个API定义了一个打印输出字符串的任务。gulp.task方法接收两个参数，第一个参数为任务的名称，第二个参数为表示具体任务的回调函数。当我们执行指定任务的时候，该任务对应的回调函数会被执行。

在终端中输入 `$ gulp demo` 指令来执行自定义任务。

```bash
GulpDemo wendingding$ gulp demo
[20:22:40] Using gulpfile ~/GulpDemo/gulpfile.js
[20:22:40] Starting 'demo'...
这是一个自定义的Task
[20:22:40] Finished 'demo' after 260 µs
```

Gulp插件Task

在Gulp作为项目构建工具使用的时候，通常主要是使用现成的相关插件来处理Task的，因为Gulp生态的原因，Gulp的插件数量足够多，质量足够好。这里我们就以一款比较流行的JavaScript代码压缩和混淆插件 `uglify` 为例简单说明。

要使用某个插件，需要先把该插件下载安装到本地。

安装插件的命令 `$ npm install --save-dev gulp-uglify`

上面的命令会把Uglify的Gulp插件下载到当前项目的node_modules文件夹下，并在package.json文件中保存相关的依赖信息。

插件安装完毕之后，我们在Gulpfile.js文件中需要先把要用到的插件引入进来，然后调用相关的API即可，下面给出代码示例。

```
//[01] 引入本地的Gulp
var gulp = require("gulp");
//[02] 引入gulp-uglify插件
var uglify = require("gulp-uglify");


//[03] 定义Task（自定义的任务）
gulp.task("demo",function(){
  console.log("这是一个自定义的Task");
})

//[04] 定义Task（插件）
gulp.task("uglifyTask",function(){
  return gulp.src("src/js/*.js").pipe(uglify()).pipe(gulp.dest("dist"));
})
```

然后通过 `$ gulp uglifyTask` 命令来执行uglifyTask任务即可，该任务执行的时候会把src/js/目录下面所有的js文件进行压缩混淆并把结果分别保存到dist目录下，下面简单列一下终端的执行情况。

BASH

```
GulpDemo wendingding$ gulp uglifyTask
[23:58:51] Using gulpfile ~/GulpDemo/gulpfile.js
[23:58:51] Starting 'uglifyTask'...
[23:58:51] Finished 'uglifyTask' after 64 ms
```

Gulp执行任务的时候，支持一次性执行多个任务，语法格式为： `$ gulp task1 task2` ，上面的Gulpfile.js文件中如果要依次执行uglifyTask和demo这两个Task，那么可以像下面这样处理。

BASH

```
GulpDemo wendingding$ gulp uglifyTask demo
[00:01:35] Using gulpfile ~/GulpDemo/gulpfile.js
[00:01:35] Starting 'uglifyTask'...
[00:01:35] Starting 'demo'...
这是一个自定义的Task
[00:01:35] Finished 'demo' after 185 µs
[00:01:35] Finished 'uglifyTask' after 65 ms
```

- Posted by 博客园·文顶顶 | 花田半亩
- 联系作者 简书·文顶顶 新浪微博·Coder_文顶顶
-