# 前后端“同构”打包

## 背景

前后端分离模式下的前端工程一般基于Vue/React天然会配置有webpack/rollup打包优化前端资源。目前4PX
前后端未分离项目占比依然很高，前端代码一般放置在Java project 下，前端代码基本处于裸奔状态。由此产生2个主要问题：

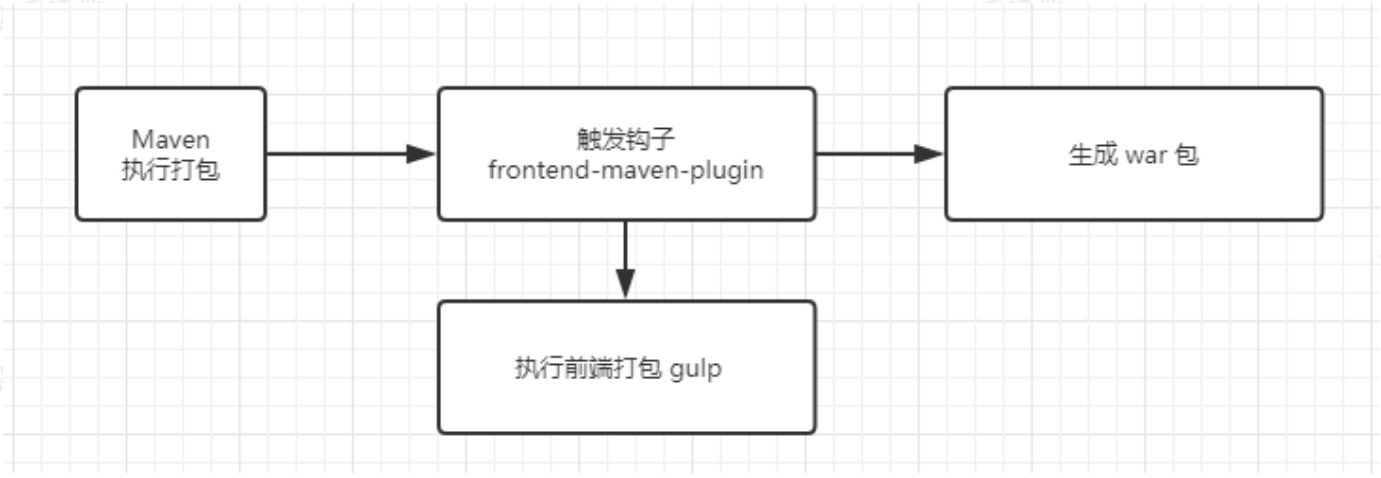
- 缓存问题，发布前手动修改版本号，发布后需要用户刷新浏览器
- 静态资源未经处理体积较大，消耗网络资源，访问较慢

## 解决方案（maven + frontend-maven-plugin + gulp）

既然maven可以打包后端，那为什么不可以连同前端一起打包了？frontend-maven-plugin插件作用就是在maven构建时可以执行gulp的打包逻辑。
https://github.com/eirslett/frontend-maven-plugin

gulp基于node强大流的能力，构建速度快，提供了丰富的插件可以满足对html/js/css/img等各种资源文件的处理。
https://www.gulpjs.com.cn/

## 具体实现




**① 配置frontend-maven-plugin插件：**

pom.xml文件配置frontend-maven-plugin，配置之后再使用maven打包会将前后端一起打包。

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.3.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo2</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo2</name>
    <description>Demo project for Spring Boot</description>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
            <exclusions>
                <exclusion>
                    <groupId>org.junit.vintage</groupId>
                    <artifactId>junit-vintage-engine</artifactId>
                </exclusion>
```

```
            </exclusions>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>com.github.eirslett</groupId>
                <artifactId>frontend-maven-plugin</artifactId>
                <version>1.9.1</version>
                <executions>
                    <execution>
                        <id>install node and npm</id>
                        <goals>
                            <goal>install-node-and-npm</goal>
                        </goals>
                    </execution>
                    <execution>
                        <id>npm install</id>
                        <goals>
                            <goal>npm</goal>
                        </goals>
                        <configuration>
                            <arguments>install</arguments>
                        </configuration>
                    </execution>
                    <execution>
                        <id>gulp build</id>
                        <phase>generate-resources</phase>
                        <goals>
                            <goal>gulp</goal>
                        </goals>
                    </execution>
                </executions>
                <configuration>
                    <nodeVersion>v12.16.1</nodeVersion>
```

```
                    <npmVersion>6.13.4</npmVersion>
                    <!-- 国内淘宝镜像-->
                    <nodeDownloadRoot>https://npm.taobao.org/mirrors/node/</nodeDownloadRoot>
                    <npmDownloadRoot>https://registry.npm.taobao.org/npm/-/</npmDownloadRoot>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

https://github.com/eirslett/frontend-maven-plugin/tree/master/frontend-maven-plugin/src/it/example%20project

**② 执行gulpfile.js中打包逻辑:**

将gulpfile.js、package.json拷贝至项目根目录下。

**打包输入**：gulp打包时需要指定源文件路径及输出路径，由于我们就项目大多基于spring boot，默认前端资源会放置在src/main/resources(static、templates)下，因此gulpfile配置的路径默认指向此，有不同请修改gulpfile.js。

**打包输出**：这里设置输出路径等于输入路径，使打包后的文件覆盖原文件。如果需要配置打包到其他路径，可在gulpfile.js中修改。
到此src/main/resource目录的HTML、JSP、CSS、JS、JPG、PNG等都已被压缩处理，HTML中引入的JS/CSS文件名称会加上hash,当文件被修改时重新打包会生成新的hash值，即合理使用缓存又保证用户不必刷新。

重要:
<!-- false: 开发时请置为false，否则会导致前端代码压缩文件名md5化 -->
<!-- true:  jenkins打包时设置 -->
var isProd = true;

```
/*
 * @Description: gulpfile.js 前端资源打包
 * @Autor: S9637/chifuk
 * @Date: 2020-09-05 11:06:59
 * @LastEditors: S9637/chifuk
```

```
 * @LastEditTime: 2020-09-07 21:23:58
 */
var gulp = require("gulp");
var uglify = require("gulp-uglify");
var pipeline = require("readable-stream").pipeline;
var babel = require("gulp-babel");
var minifyCss = require("gulp-minify-css");
var imagemin = require("gulp-imagemin");
var pngquant = require("imagemin-pngquant");
var rev = require("gulp-rev");
var revCollector = require("gulp-rev-collector");
// var clean = require("gulp-clean");
var gulpif = require("gulp-if");
var del = require("del");
var vinylPaths = require("vinyl-paths");
var htmlMinify = require("gulp-htmlmin");

<!-- false: 开发时请置为false，否则会导致前端代码压缩文件名md5化 -->
<!-- true:  jenkins打包时设置 -->
var isProd = true;

// 输入输出目录
var publish = {
    root: "src/main/resources",
    static: "src/main/resources/static",
    templates: "src/main/resources/templates",
};
// var sourceFiles = ["static/**", "templates/**"];

// 映射文件输出目录
var manifestPath = "src/main/resources/manifest";

// 1.打包前清空文件
// gulp.task("clean", function () {
//     return pipeline(gulp.src([publish.root], { allowEmpty: tr
ue }), clean());
// });

// 2.将源代码拷贝至打包目录
// gulp.task("copy", function () {
```

```
//     return pipeline(gulp.src(sourceFiles), gulp.dest([publish.static, publish.templates]));
// });

// 3.文件名 hash 化，解决缓存问题
// 设置 JS/CSS/HTML 文件映射
gulp.task("revJsCss", function () {
    return pipeline(
        gulp.src([publish.static + "/**/*.js", publish.static + "/**/*.css"]),
        vinylPaths(del),
        rev(),
        gulp.dest(publish.static),
        rev.manifest(),
        gulp.dest(manifestPath)
    );
});

// 映射文件注入 HTML
gulp.task("revCollector", function () {
    return pipeline(
        gulp.src([manifestPath + "/*.json", publish.templates + "/**/*.html"]),
        revCollector({
            replaceReved: true,
        }),
        gulp.dest(publish.templates)
    );
});

// 4.资源压缩处理（html/js/css/img..）
gulp.task("htmlMin", function () {
    return pipeline(
        gulp.src(publish.templates + "/**/*.html"),
        htmlMinify({
            removeComments: true,
            collapseWhitespace: true,
            minifyJS: true,
            minifyCSS: true,
        }),
```

```
        gulp.dest(publish.templates)
    );
});

gulp.task("cssMin", function () {
    return pipeline(
        gulp.src(publish.static + "/**/*.css"),
        minifyCss(),
        gulp.dest(publish.static)
    );
});

// 排除已压缩文件
var condiction = function (f) {
    if (f.path.endsWith(".min.js")) {
        return false;
    }
    return true;
};

gulp.task("jsMin", function () {
    return pipeline(
        gulp.src(publish.static + "/**/*.js"),
        babel({
            presets: ["@babel/env"],
        }),
        gulpif(condiction, uglify()),
        gulp.dest(publish.static)
    );
});

gulp.task("imgMin", function () {
    return pipeline(
        gulp.src(publish.static + "/**"),
        imagemin({
            progressive: true,
            use: [pngquant()],
        }),
        gulp.dest(publish.static)
    );
```

```
});

/**
 * 开发时，将isProd = false
 * 因为默认打包的的文件会覆盖原文件，所以仅在jenkis打包环境设置为true
 */
if (isProd) {

    gulp.task(
        "default",
        gulp.series(
            // "clean",
            // "copy",
            "revJsCss",
            "revCollector",
            gulp.parallel("htmlMin", "cssMin", "jsMin", "imgMin")
        )
    );
}else {
    console.log('开发环境，不执行 gulp 打包，明智的选择！')
}
```

```
{
  "name": "gulp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@babel/core": "^7.11.6",
    "@babel/preset-env": "^7.11.5",
    "gulp": "^4.0.2",
    "gulp-babel": "^8.0.0",
```

```
    "gulp-clean": "^0.4.0",
    "gulp-htmlmin": "^5.0.1",
    "gulp-if": "^3.0.0",
    "gulp-imagemin": "^7.1.0",
    "gulp-minify-css": "^1.2.4",
    "gulp-rev": "^9.0.0",
    "gulp-rev-collector": "^1.3.3",
    "gulp-uglify": "^3.0.2",
    "gulp4-run-sequence": "^1.0.1",
    "imagemin-pngquant": "^9.0.0",
    "readable-stream": "^3.6.0",
    "vinyl-paths": "^3.0.1",
    "del": "^5.1.0"
  }
}
```

## 注意事项

- gulp执行环境为node,首次打包会下载依赖生成node_modules，次依赖文件较大，请配置各种ignore（svn/git）避免引用；
- Intellig Idea、ecplice设置忽略node_modules；
- 开发环境，配置gulpfile.js中 isProd=false;