课窝技术开发规范

开发部分

课窝技术部

2018.10

一、概述

为提升多人协作效率，保证项目开发质量，方便后期维护，结合公司技术开发实际情况，特编写此规范。开发人员需认真阅读此规范，在实际开发中加以注意；以便写出高质量、少往复、便协作的代码。

项目采用 laravel 框架，按照现有服务器运行情况，使用 V5.3 或者 V5.4 版本，个别项目可采用 5.5 以上版本。PHP 版本为 5.6。

二、项目初始化

2.1 结构划分

控制器、Model、视图及资源等需要按照模块来划分，分别建立对应的文件夹，上传图片统一存储于 public/uploads/temp 临时文件夹中，在真正存储的时候转移至对应的文件夹下。

2.2 入口

项目入口设置为 public 下的 index.php,以便保护根目录下其他文件。

2.3 AUTH 认证

中间件的用处很多，最主要的也是必须要设置的两个事 VerifyCsrfToken 和 AdminAuth，分别管理 CDRF 的验证和后台用户的验证及权限。Csrf 中间件可以设置例外路由，不用参与验证码，我们在开发过程中，除非必要一般不设置该项。

2.4 中间件

后台用户的验证使用 auth 类来验证，自定义 guard 为 admin,权限使用 Gates。具体使用方法请参考相关官方文档。

2.5 代码管理

使用 git 作为代码管理工具，在 git 提交代码时，必须忽略掉工程配置文件，比如（nbproject 下的文件），vendor 和.env 文件也需要忽略掉，但是 env 中的参数必须在.env.example 中写明，该文件可以命名为“.env.开发者名”。

2.6 数据库

在开发阶段，如果修改过数据库，一定要在 database 文件夹下导出一份数据库完整 sql 文件；在维护阶段，不允许删除表或者字段，在新增表或者字段的时候，必须单独以文件的形式列出所有执行过 sql 语句，该文件命名为“数据表修改 2018-xx-xx（未更新）.txt”，同时导出一份最新的完整的数据表，以“databaseName2018-xx-xx.sql”命名，databaseName 为实际数据库名。

2.7 composer

在 composer 安装类库的时候，一定要注意自己本地环境变量中的 php 版本，要符合 composer 中对 php 的最低要求，即 Laravel5.3 版本，必须在本地 php 环境是 5.6 的情况下才可以 install，否则会出错!

三、开发规范

在实际开发过程中，我们一定要按照如下约定进行开发，方便调试与维护。

1. 命名规范

1.1 文件夹命名

文件夹命名以贴合实际意义的英文单词为主，个别可以使用拼音。
使用大驼峰法命名，控制器、Models、异常、中间件等处首字母大写，视图、公用部分等处完全使用小写命名。
例如：后台管理控制器文件夹路径为：app/Http/Controllers/Manager/
视图文件夹路径为：resources/views/manager

1.2 控制器命名

控制器位置在 App/Http/Controllers。

1.2.1 控制器文件名

控制器命名以对用操作数据表名或者实际意思英文单词命名，单词后紧跟 Controller，并以.php 为后缀。
文件命名使用大驼峰法，首字母大写。
控制器类的命名空间要和改文件所在的路径保持一致。
例如：基控制器 BaseController.php
命名空间 namespace App\Http\Controllers\Manager;

1.2.2 控制器类名

控制器类名的命名与该类的文件名保持一致，命名后要加 Controller。
例如：基控制类：class BaseController { }

1.2.3 控制器方法名

控制器方法名需要明确表述该方法的实际意义，使用小驼峰法，并且首字母小写或者使用下划线“_”，通常下划线开头的方法属于私有方法。
例如：获取用户名 getUserName(){}

1.3 视图命名

视图文件夹路径为：resources/views。

1.3.1 视图文件夹命名

首先需要根据前后台或是控制器的模块来建立对应的视图文件夹，命名取控制器文件夹名，小写；该文件夹下需要建立布局文件夹 layout，用以存放视图公有布局文件；然后根据模块下控制器类名的小写形式，建立文件夹，存放当前控制器中的方法所用到的视图文件。
例如：后台模板路径：resources/views/manager。
后台布局路径：resources/views/manager/layout
后台用户操作相关：resources/views/manager/user

1.3.2 视图文件命名

视图文件的命名一般取对应的控制器中方法名，单独的命名需要登表明该视图的作用。
文件名是小写形式，以“.blade.php”结尾。
特定的几个文件命名：基础布局（base.blade.php）、样式（head.blade.php）、头部（header.blade.php）、尾部（footer.blade.php）、右边栏（rsidebar.blade.php）、首页（index.blade.php）、新增（create.blade.php）、编辑（edit.blade.php）、详情（show.blade.php）、文档（document.blade.php）、登录（login.blade.php）……

1.4 助手函数命名

助手函数命名同样需要以表明实际功能为原则，取英文名。
命名使用蛇形命名法，由小写字母和下划线组成。
例如： get_client_ip()

1.5 js 命名

js 文件存放于对应以模块名命名的公用文件加下。

1.5.1js 文件名

首先公用 js 代码存放于 common.js 文件中，其他页面用到的 js 可以使用当前页面的视图名来命名，或者取当前 js 文件主要功能描述性词汇。

以小驼峰法命名。

例如：userList.js

1.5.2js 方法名

方法名以简单准确为原则命名，使用小驼峰法。

例如：var ageValue;

1.6 CSS 命名

CSS 文件存放于对应以模块名命名的公用文件加下。

1.6.1CSS 文件名

首先重置样式文件命名为 reset.css，公用样式文件命名为 common.css，其他 css 文件可以使用当前页面的视图名来命名，并且全部小写。

例如：laydate.css

1.6.2CSS 样式名

样式名要能体现改样式所控制的标签及状态，命名采用脊柱命名法，全部小写。

例如：.layui-form-checked

1.7 图片命名

公用图片、icon 等存放于对应以模块名命名的公用文件加下，上传的图片存放于 public/uploads/模块名/分类

1.7.1 公用图片命名

公用图片的命名需要表明图片的用途内容，采用脊柱命名法，全部小写。
例如：icon-book.png

1.7.2 上传图片命名

上传的图片存放在 public/uploads/模块名下边，各个部分用到的图片使用控制器名、数据表名或者实际意义名，然后该文件夹下的图片以当天日期为文件夹名存储，图片名称进行 MD5 加密。
同时建立 public/uploads/temp 文件夹，用以存放临时文件，后台在上传图片操作是，首先传至该临时文件，在真正执行存储过程中在做图片迁移。
例如：/public/uploads/manager/2018-11-29/5djrrj582jd7exngsp42j07azigm4dk.png

1.8 数据库命名

数据库命名以网站域名为主，或者以实际意义来命名，采用蛇形命名法，字母小写。
例如：kewo_manager　live_kewo_com

1.8.1 数据表命名

数据表名的命名以实际表的作用为主，需要添加前缀，用下划线隔开，尽量使用名词复数形式，表明全部小写。
多对多的表使用两个表命名的单数形式，然后采用蛇形命名法。
例如：kewo_users kewo_roles　kewo_user_role

1.8.2 字段命名

数据表字段尽量保持一个单词，单数形式，采用蛇形命名法，小写。
固定几个名词 id（ID）、username（用户名）、nickname（用户昵称）、标题（title）、关键字（keywords）、电话（phone）、邮箱（email）……

1.8.3Model 命名

模型的命名采用数据库名单数形式，大驼峰法命名，首字母大写，以“.php”结尾。文件位置 app/Models/模块名。公用部分可以直接在 Models 下面。
例如：app/Models/User.php

1.8.4Model 方法命名

模型下的方法命名采用小驼峰法，要能清楚表达方法意义。
例如：getUsername()'

1.9 其他

1.9.1PHP 变量命名

属性或者变量的命名一定要表明该属性的实际意义，禁止使用$a $b 等这样的命名，如果命名无法表明意义，或者是有单独的含义，需要在该变量或则属性之后做注释。
命名使用驼峰法，并且首字母小写或者使用下划线“_”，通常下划线开头的方法属于私有属性。
例如：用户名：$userName

1.9.2 JS 变量命名

在开发中，所有的变量必须在使用前声明好，每个变量单独成一行，并在其后加以注释，顺序以字母排序。
采用驼峰法，首字母小写。
例如：var ageValue， //用于存诸年龄值
username； //用户名

1.9.3 路由命名

路由命名基本与控制器的方法名一致，可以根据 SEO 要求来定制，路由有字母和“_”组成，在路由文件中需要添加路由别名，别名的命名方式为 模型名+控制器名+方法名
例如：文章详情 http://www.xxx.com/article_6.html 别名为 index.article.show

2. 数据库规范

2.1 字段

字段的命名已经提过，命名时一定要能准确表述该字段的含义。另外在新增字段的时候，需要设置默认值。

2.2 注释

在设计表的时候，每个字段都需要注释，以便日后 review 时能明白该字段的含义。该数据表也需要注释，注

明是什么表，有什么用。

2.3 其他

表之间涉及关联的，一定要建立索引和外键，外键删除时用 CASCADE 更新时用 RESTRICT，当然并非都要这么设置，根据实际需要设置。

表的索引选择 BTREE 方式

表的引擎选用 InnoDB

字符集：utf8 -- UTF-8 Unicode

排序规则：utf8_general_ci

3. 代码规范

3.1 注释

最基础也是最重要的一部分就是注释，千万不要吝啬注释。当然也需要合理注释。注释这一栏单独列出来，就是因为它很重要。

开发者最烦的有两件事，第一件事是别人要他给自己的代码写文档，第二件呢是别人的程序没有留下文档。注释也是一样的，当我们看到很多绿色的注释代码的时候神经会比较放松，而看到揉成一团还没有注释的代码是相当压抑的。

3.1.1 PHP 类注释

在每个类开始的地方，构造函数之上，需要添加类注释。

```
/*
|--------------------------------------------------------------------------
| Ad Controller
|--------------------------------------------------------------------------
|
| 广告控制器
|
*/
```

用以表明该类属于什么类，作用是什么。

3.1.2 PHP 方法注释

在方法名上边，需要添加方法的注释，格式如下：

```
/**
```

```
    * 这里填写该方法的作用
    *
    * @param string p 当前页数 默认 1
    * @param string limit 每页数量 默认 10
    * @param user_id int 用户 ID
    * @return array $return
    */
    public function updateWordlist() {}
```

需要注明当前方法所需要的参数，并注明类型、参数名、意义、取值、是否必须等信息，然后注明该方法返回值的类型、变量名及返回值的含义。

3.1.3 PHP 变量语句注释

变量的注释比较简单，于当前变量后双斜杠注释即可：

```
$return['errno'] = 1203; //返回值
```

语句的注释同样，只需在语句上面注释，每一段功能的语句执行完，需要空一行：

```
//输入值获取
$type = $request->input('type',2);
$limit = $request->input('limit',100);

//获取广告数据
$ads = Ad::get()->toArray();
```

3.2 PHP 代码

代码在写的时候一定要格式化，符合共同约定。

3.2.1 基本约定

a. 源文件

a) .纯 PHP 代码源文件只使用 <?php 标签，省略关闭标签 ?> ；
b) 源文件中 PHP 代码的编码格式必须是无 BOM 的 UTF-8 格式；
c) 使用 Unix LF(换行符)作为行结束符；
d) 一个源文件只做一种类型的声明，即，这个文件专门用来声明 Class，那个文件专门用来设置配置信息，别混在一起写；

b. 缩进

使用 Tab 键来缩进，每个 Tab 键长度设置为 4 个空格；

c. 行

一行推荐的是最多写 120 个字符，多于这个字符就应该换行了，一般的编辑器是可以设置的。

d. 关键字和 TRUE/FALSE/NULL

PHP 的关键字，必须小写，boolean 值：true，false，null 也必须小写。
下面是 PHP 的“关键字”，必须小写：

```
'__halt_compiler', 'abstract', 'and', 'array', 'as',
'break', 'callable', 'case', 'catch', 'class', 'clone',
'const', 'continue', 'declare', 'default', 'die', 'do',
'echo', 'else', 'elseif', 'empty', 'enddeclare', 'endfor',
'endforeach', 'endif', 'endswitch', 'endwhile', 'eval',
'exit', 'extends', 'final', 'for', 'foreach', 'function',
'global', 'goto', 'if', 'implements', 'include', 'include_once',
'instanceof', 'insteadof', 'interface', 'isset', 'list',
'namespace', 'new', 'or', 'print', 'private', 'protected',
'public', 'require', 'require_once', 'return', 'static',
'switch', 'throw', 'trait', 'try', 'unset', 'use', 'var',
'while', 'xor'
```

e. 命名

所有类名方法名按照上节的命名规范来命名。

f. 代码注释标签

如 函数注释、变量注释等，常用标签有 @package、@var、@param、@return、@author、@todo、@throws
必须遵守 phpDocument 标签规则，不要另外去创造新的标签，更多标签查看 phpDocument 官网

g. 业务模块

a) 涉及到多个数据表 更新/添加 操作时，最外层要用事务，保证数据库操作的原子性；
b) 业务逻辑统一封装到 Controller 层；
c) 数据处理统一到 Model 中

3.2.2 代码样式风格

a. 命名空间(Namespace) 和 导入(Use)声明

a) 命名空间(namespace)的声明后面必须有一行空行；
b) 所有的导入(use)声明必须放在命名空间(namespace)声明的下面；

c) Use 门面或者其他依赖放在一起，然后空一行放 use Model;
d) 一句声明中，必须只有一个导入(use)关键字;
e) 在导入(use)声明代码块后面必须有一行空行;
例如:

```php
<?php

namespace App\Http\Controllers\Api;// 下面必须空格一行

use Illuminate\Http\Request;// use 必须在 namespace 后面声明
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Storage;
use App\Http\Controllers\Api\TokenController;// 下面必须空格一行

use App\Models\Category;
use App\Models\Ad;// 下面必须空格一行

class AdController extends TokenController {}
```

b. 类(class)，属性(property)和方法(method)

a) 类名和继承必须写一行
b) 属性(property)必须声明其可见性，到底是 public 还是 protected 还是 private，不能省略，也不能使用 var, var 是 php 老版本中的什么方式，等用于 public。
c) 方法(method)，必须 声明其可见性，到底是 public 还是 protected 还是 private，不能省略。如果有多个参数，第一个参数后紧接“,”，再加一个空格：function_name ($par, $par2, $pa3)，如果参数有默认值，“=”左右各有一个空格分开。
例如:

```php
class AdController extends TokenController {
private $name = 'yangyi';

// 参数之间空格。默认参数的“=”左右各有一个空格，) 与 { 之间有一个空格
public getInfo($name, $age, $gender = 1) { }
```

c. 控制结构

控制接口，就是 if else while switch 等。这一类的写法规范也是经常容易出现问题的，也要规范一下。
a) if，elseif，else 写法，直接上规范代码吧:

```php
<?php
if ($expr1) { // if 与 ( 之间有一个空格，) 与 { 之间有一个空格

} elseif ($expr2){ // elesif 连着写，与 ( 之间有一个空格，) 与 { 之间有一个空格

} else { // else 左右各一个空格
```

```
7
8 }
```

b) switch，case 注意空格和换行，还是直接上规范代码：

```php
<?php
switch ($expr) { // switch 与 ( 之间有一个空格，) 与 { 之间有一个空格
    case 0:
        echo 'First case, with a break'; // 对齐
        break; // 换行写 break，也对齐。
    case 1:
        echo 'Second case, which falls through';
        // no break
    case 2:
    case 3:
    case 4:
        echo 'Third case, return instead of break';
        return;
    default:
        echo 'Default case';
        break;
}
```

c) while，do while 的写法也是类似，上代码：

```php
<?php
while ($expr) { // while 与 ( 之间有一个空格，  ) 与 { 之间有一个空格

}

do { // do 与 { 之间有一个空格

} while ($expr); // while 左右各有一个空格
```

d) for 的写法

```php
<?php
for ($i = 0; $i < 10; $i++) { // for 与 ( 之间有一个空格，二元操作符 "="、"<" 左右各有一个空格，) 与 { 之间有一个空格

}
```

e) foreach 的写法

```php
<?php
```

```
foreach ($iterable as $key => $value){ // foreach 与 ( 之间有一个空格，"=>" 左右各有一个空格，) 与 { 之间有一个空格

}
```

f) try catch 的写法

```
<?php
try { // try 右边有一个空格

} catch (FirstExceptionType $e) { // catch 与 ( 之间有一个空格，) 与 { 之间有一个空格

} catch (OtherExceptionType $e) { // catch 与 ( 之间有一个空格，) 与 { 之间有一个空格


}
```

d. 注释

注释按照上讲的说明执行。

e. 空格

赋值操作符（=，+= 等）、逻辑操作符（&&，||）、等号操作符（==，!=）、关系运算符（<，>，<=，>=）、按位操作符（&，|，^）、连接符（.） 左右各有一个空格。

if，else，elseif，while，do，switch，for，foreach，try，catch，finally 等 与 紧挨的左括号"("之间有一个空格。

函数、方法的各个参数之间，逗号（","）后面有一个空格；

f. 空行。

所有左花括号 { 都不换行，并且 { 紧挨着的下方，一定不是空行；

同级代码（缩进相同）的 注释（行注释/块注释）前面，必须有一个空行；

各个方法/函数 之间有一个空行；

namespace 语句、use 语句、clase 语句 之间有一个空行；

如果 return 语句之前只有一行 PHP 代码，return 语句之前不需要空行；

如果 return 语句之前有至少二行 PHP 代码，return 语句之前加一个空行；

if，while，switch，for，foreach、try 等代码块之间 以及 与其他代码之间有一个空行。

g. 数组的书写格式

```
$where = array('id' => 789); //只有一个值时
```

```
//有多个值时
$where = array(
```

```
    'id' => 789,
    'user_name' => '52php'
);
```

书写原则：做到 代码紧凑 而又不失 小模块化 ！

3.2.3 PSR-4 规范

PSR-4 规范是刚出没多久的一条新的规范，它也是规范 自动加载(autoload)的，是对 PSR-0 的修改，属于补充规范，

3.3 HTML 代码

3.3.1 HTML 代码规范

在开发过程中一定要注意行级元素、块计元素区别使用。正确闭合标签且必须闭合，空行缩进注释都要得当，属性和值全部小写，每个属性都必须有一个值，每个值必须加双引号。具体更详细的一些约定，请阅读《课窝技术开发规范 1.1》这篇文档。

a. 基础部分

a) 文件应以<!DOCTYPE.....>首行顶格开始；
b) 必须在 head 元素内部的 meta 标签内声明文档的字符编码 charset，如：<meta charset="UTF-8">；
c) 页面的 title 是极为重要的不可缺少的一项。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>

</body>
</html>
```

b. 顺序

a) 按照从上之下，从左到右的视觉顺序书写 HTML 结构；
b) 为了便于搜索引擎抓取，可以将部分重要结构提前；

c. 结构表现行为

a) 使用 link 引入外部 css 文件到 head 中
b) 使用 script 将 js 文件引入，并置于 body 底部
c) Html 页面不要写过多的 css 样式和 js 代码

d. 树形结构

保持良好的树形结构，每一个块级元素都另起一行，每一行都是用 tab 缩进，注意：html、 head、 body 以及 body 下的第 1 级标签（即直接子元素）不缩进，其他的都正常缩进。

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
<h1></h1>
</body>
</html>
```

e. 其他

a) 一个标签上引用的 className 不要过多，越少越好。
b) 对于一个语义化的内部标签，应该尽量避免使用 className。

f. 注释

不要吝啬注释，但也要合理注释，每一个功能结构都需要一个文案注释，
<!-- 注释文案 --> <!-- /注释文案 -->

3.3.2 SEO 规范

开发 SEO 友好网站，前端开发需要注意一下几点：合理的标签布局、文章排版、页面排版等等。

a. TDK

每个页面必须写 TDK,<title></title>，description，keywords，名称要写对，注意使用英文引号。

b. 样式

样式尽量写在 css 文件中，如果需要写内部样式，代码需写在<head>标签中间。

c. <h#>标签

一般情况下，搜索引擎蜘蛛会分析页面的结构，对于文章页面来说，<h1\>，<h2\>，<h3\>等标题标签具有不同的权重，另外<b\>，<i\>，<strong\>等具有强调作用的标签也会有更高的权重。
<h1\>标签只在页面中出现一次，一般用于文章标题，至于字体的大小，请使用 css 控制。

d. <img>标签

此标签必须添加 alt 属性，一般为栏目名称、标题名、关键词，图片必须限制尺寸，需锁死宽度，高度可设置一个最高高度。

e. 文章详情

在网站后台编辑文章的时候一般是由富文本编译器直接编辑的，所以前端在开发页面的时候需要考虑这方面的情况，如下是一篇文章的简单示例，可以在开发文章详情页的时候直接引入到对应的地方，以便之后直接使用不会有样式问题。

```
<p style="line-height: 2em;">
    <strong>
        <span style="font-family: 微软雅黑,Microsoft YaHei;">    留学申请需匹配兴趣及强项
        </span>
    </strong>
</p>
<p style="line-height: 2em;">
    <span style="font-family: 微软雅黑,Microsoft YaHei;">    在选择专业时，你也应该多少考虑到未来 30 年里你会做什么样的工作。要想拥有成功的事业</span>
</p>
<p style="line-height: 2em;">
    <strong>
        <span style="font-family: 微软雅黑,Microsoft YaHei;">    澳洲留学语言环境需适应</span>
```

```
        </strong>

    </p>

    <p style="line-height: 2em;">

        <span style="font-family: 微软雅黑,Microsoft YaHei;">    澳洲留学申请语言环境是选择专业时应考虑的重要因素。大部分中国大陆学生选择在澳洲学习商科，工科以及 IT 领域。</span>

    </p>

    <p><br></p>
```

3.3.3 Balde 模板

在使用框架开发的时候，我们需要将前端页面改造为 blade 模板，在进行页面切割的时候，一定要注意标签的闭合，不可出现切割后，个别标签的闭合问题导致的 html 结构混乱。

在项目中尽量保持如下的模板分割和继承，命名需注意：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>KeWo</title>
    <meta name="renderer" content="webkit">
    @yield('head')
</head>
<body class="childrenBody">
<header>
    @yield('header')
</header>

<!--正文开始-->
<div>
    @yield('content')
<div>
<!--/正文结束-->
```

```
<footer>
    @yield('footer')
</footer>
</body>
```

```
    @yield('script')
    </html>
```

3.4 JS 代码

3.4.1 基础

a. 命名及注释

命名和注释参考上节的一些规范。全局变量和常量使用大写形式。

b. 空格与运算符

通常运算符（= + - * /）前后需要添加空格。为了便于阅读每行字符建议小于数 80 个。如果一个 JavaScript 语句超过了 80 个字符，建议在 运算符或者逗号后换行。

c. 语句规则

每条语句必须以分号作为结束符

3.4.2 项目开发

些简单的或者必要的代码可以写在页内，如果涉及到比较复杂的功能或者其他数据处理，需要单独建立该页面的 js 文件，多个页面共同的一些功能需要单独写成 common 文件，方便调用，如 form 格式判断。

3.4.3 引入

使用简介格式载入 JavaScript 文件，在模板中，每个页面单独使用的 js 文件不要在基模板中引入，避免 js 不同导致的的页面错误。

3.5 CSS 代码

3.5.1 基础

a. 命名及注释

命名和注释参考上节的一些规范，一定要多写注释，

b. 书写顺序

属性的书写顺序对于浏览器来说没有区别，除了优先级覆盖之外。但是如果顺序保持一致的话，扫一眼可以很快地知道这个选择器有什么类型的属性影响了它，所以一般要把比较重要的属性放前面：

Display/float/postion > 盒模型的属性>文本属性>CSS3 的一些属性

c. 其他

a) 不要使用样式特点命名
例如 red-font ，加入 UI 改了颜色，那这个样式就没意义了。

b) 选择器的性能
选择器一般不要超过 3 个，不用每个容器都写进去，重要的目标元素套上 class 或者 ID。

c) 全局设置
一般不要写全局选择匹配器，或者是全局改变标签的属性，尤其是 span 标签。

d) 属性值
如果值是 0，通常都不用带单位；色值用十六进制，少用 rgb；注意 border none 和 0 的区别。

e) 设置常见样式 reset
由于每个浏览器都有自己的 UA 样式，并且这些样式还不太统一，所以需要做样式 reset。

f) 图片压缩
不管是 UI 直接给的图片还是自己从 UI 图里切出来的图片，都需要把图片压缩一下，建议使用 tinypng，它可以在保持图片质量减少较低的情况下，把图片压得很厉害，比直接在 PS 里面设置压缩质量要强。如果是色彩比较丰富的图片要使用 jpg 格式，不能使用 png 格式，png 会大得多，如果是 logo 那种矢量图片，直接使用 svg 格式即可。一般来说要把图片控制在 300k 以内，特别是 banner 头图，图片的大小也要控制住。

3.5.2 项目开发

在实际项目中，样式尽量写在样式文件中，行内样式除非必要，否则一律写 ClassName。

四、其他

在项目开发过程中，开发人员需要每天提交两次代码。在划分的每个小功能结束时，需要同小组长检查一遍代码及功能；所有功能开发结束后，需要自己先检查一遍项目。

在项目开发结束前，开发人员需要将代码 git 提交上传，或者是压缩发送给负责人，由项目负责人上传至测试服务器供大家测试。

测试组成员主要有：产品经理、前端开发、后端开发。

测试需要注意的几个方面：功能实现、代码规范、浏览器兼容、文字超出处理、图片样式、动效等等。