给一个具体的东西命名时，能用一个单词表达清楚意思，就不要使用多个单词组合，如果光靠字面意思去匹配单词，那么大多数都是单词组合，建议去搜索一下对应的名词单词表示方式！

如果 文件2 实现的功能是属于 文件1 中，那么我称 文件1 为 文件2 的父级，文件2的命名方式为：

文件1的名字 + 文件2实现的功能名字。（可以理解为 父级名字 为 子级名字的前缀）

ex: 比如说我有一个父级表( session )，session里面字段的文字是静态的、图片是动态的，那么我为session分为两个表，session-content表 和 session-img 表。

目录名字、文件名字命名规范：多个单词组成的目录名字、文件名字使用 - 分隔。

数据库表名字的命名规范：多个单词组成的表名使用 \_ 分隔。

根目录单词如果有复数，使用复数形式定义目录名字。

定义表文件名字规范： 首字母大写 （Project.js）

数据库表名字的命名规范：

表的名字统一使用小写，多个英文单词组成必须用 \_ 分割。

根表下面的子表定义格式： 必须在前面带上所有上级表的名字，用 \_ 分隔。

例如：

子表格式：basic表的上级表是projects， 那么basic表的名字应该是这样：

projects\_basic

子表下的子表格式：person表的上级表是basic，那么person表的名字应该是这样：

projects\_basics\_person

**前后台的路由和控制器的定义规范一致**，如下所示：

路由的定义规范：

路由文件的名字统一使用小写。

根路由下面的子路由必须定义在根路由文件下。

根路由下面的子路由定义格式： 必须在前面带上所有根路由的名字，用 / 分隔。

路由接口的命名规范：使用名词命名，而不是使用动词命名。

一个很重要的软件开发原则：针对接口开发，而不是针对实现开发。这个原则好处在于：我们能把静态的东西封装在接口中，便于实现的动态扩展。

例如：

projects.js 是 定义 根路由为 /projects 的文件，所有的 /projects 下面的 子路由必须定义在该文件下。

module.exports = {

path: "/projects", //根路由

route: router

};

根路由格式： router.get(‘/’, projects.get);

子路由格式： router.get(‘/basic’, basic.get);

子路由下面的子路由格式： router.get(‘/basics/person’, person.get);

如果要为当前接口实现特定的功能，统一定义：

例如：忘记密码：

router.post(‘/:id/password?action=forget’);

前边加操作，后边加上动作，增强可读性。

//router.get(‘/config/menu-item’);

//添加/config表示接下来的接口是属于我这个根接口配置的特定功能。

控制器的定义规范：

控制器文件的名字统一使用小写。

控制器文件定义的名字 要与 路由文件的名字一致。

（router/projects.js -> controller/projects.js）

如果控制器projects.js下面有多级子控制器，那么projects必须定义成一个文件夹，projects下面的所有子控制器必须定义在projects文件夹下。

子控制器的位置：projects/basic.js

子控制器下的子控制器的位置 projects/basics/person.js

凡是以 - 分隔的目录名、文件名、接口名等，对应的调用方式统一把 - 变成 \_ 。(去横线变下划线)

注释推荐使用块注释：(S start 开始、E end 结束)

/\*信息 S\*/

代码

/\*信息 E\*/