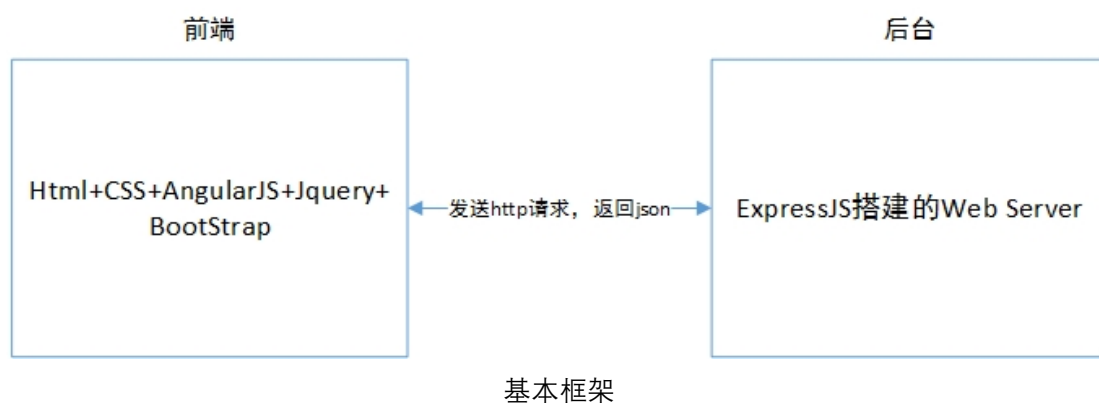


说明文档

1. 简介

本项目旨在理解并使用初级的 AngularJS 技术及 NodeJS 技术的 Express 框架实现一个简单的只读性网站。AngularJS 的路由机制为单页面应用提供了可能，它能够向单一页面中注入不同的内容，从而表现出多页面的样式，但同时它能够对页面的访问记录进行缓存，从而使得返回变的更加容易。Express 框架提供了一个清晰的结构，能够简洁快速的实现对前端请求的获取与回应。

由于学习的时间较短，内容较浅，因此本项目只实现了最基本的路由，和简单的数据交互，相关技术将在下文说明。



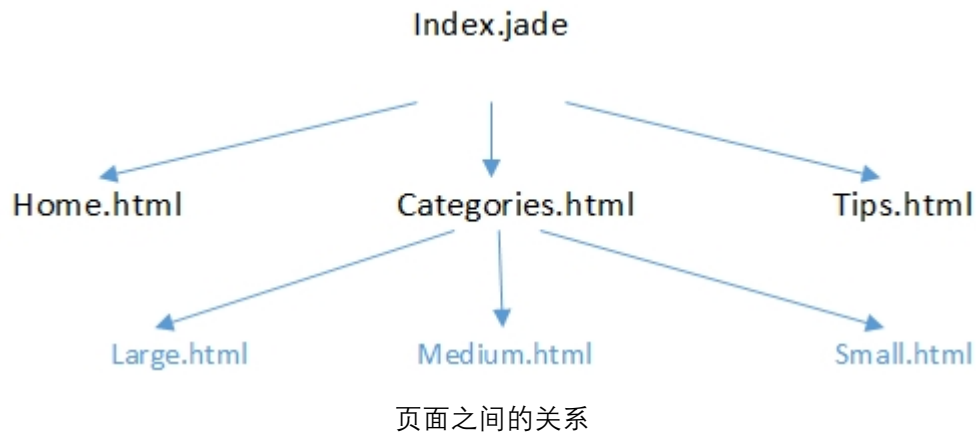
2. 技术说明

2.1. AngularJS 路由机制

AngularJS 的原生路由可以解决简单的多页面切换，但是 ui-router 更加适合用来表达路由之间的某些继承关系。ui-router 是一个可以用来替换 AngularJS 原生路由的插件，它最大的特点就是支持嵌套路由，本项目使用 ui-router 实现了一个简单的三层关系的页面。

Index.jade 页面由导航栏和展示区组成，其中展示区依赖路由获取对应页面

的内容注入到容器中（ui-view 对应的 div 中），从而在一个页面中通过切换导航获取不同的显示内容。Categories.html 中有三个 tab，通过切换 tab 在 categories 页面中又划分了三个子页面，其机制与上一层一致。



原生路由在本项目中也有尝试：在这种情况下，通过 when（）方法来定位不同路由对应的 url，获取相应页面，但是对于页面中子页面的展示，ui-router 则更加方便。

```
var myApp = angular.module('myApp', ['ui.router', 'myApp.controllers', 'categories.controllers']);

//myApp.config(function($routeProvider) {
//  $routeProvider
//  .when('/home', {
//    templateUrl : 'templates/home.html',
//    controller : 'homeCtrl'
//  })
//  .when('/categories',{
//    templateUrl : 'templates/categories.html',
//    controller : 'categoriesCtrl'
//  })
//  .when('/tips',{
//    templateUrl : 'templates/tips.html',
//    controller : 'tipsCtrl'
//  })
//  .otherwise({redirectTo: '/home'})
//});
```

原生路由实现页面切换

ui-router 可以定义子路由，实现了分页面中嵌套多个页面的功能，根据定义 state，来说明路由之间的关系，其中 categories.large、categories.medium、

categories.small 为 categories 的子路由。

```
myApp.config(['$stateProvider', '$urlRouterProvider', function ( $stateProvider, $urlRouterProvider ) {
    $urlRouterProvider.when('', '/home');
    $stateProvider
        .state('home', {
            url: '/home',
            templateUrl: 'templates/home.html',
            controller: 'homeCtrl'
        })
        .state('categories', {
            url: '/categories',
            abstract: true,
            templateUrl: 'templates/categories.html'
        })
        .state('categories.large',{
            url: '',
            templateUrl: 'templates/large.html',
            controller: 'largeCatCtrl'
        })
        .state('categories.medium',{
            url: '/medium',
            templateUrl: 'templates/medium.html',
            controller: 'mediumCatCtrl'
        })
        .state('categories.small',{
            url: '/small',
            templateUrl: 'templates/small.html',
            controller: 'smallCatCtrl'
        })
        .state('tips', {
            url: '/tips',
            templateUrl: 'templates/tips.html',
            controller: 'tipsCtrl'
        })
    });
```

ui-router 实现页面嵌套 (public/js/app.js)

在 public/js/app.js 文件中定义了路由和每个页面对应的控制器，在 public/js/controllers.js 中定义了每个 controller 的 scope，其中主页和贴士页面的展示内容为直接在 controller 中定义的文本，另一部分则通过使用 AngularJS 的 \$http 请求向服务器发送请求获取数据，相应页面加载时自动向服务器发送请求，获取的数据则在前台的 html 文件中进行渲染。

```

angular.module('categories.controllers', [])

// 分类页信息内容，页面加载时向后台发起请求,获取json数据

.controller('largeCatCtrl', function($scope, $http){
    $scope.getLarge = function(){
        $http.get('http://localhost:3000/getLarge').success(function(res) {
            $scope.cats = res;
        });
    };
    $scope.getLarge();
})
.controller('mediumCatCtrl', function($scope, $http){
    $scope.getMedium = function(){
        $http.get('http://localhost:3000/getMedium').success(function(res) {
            $scope.cats = res;
        });
    };
    $scope.getMedium();
})
.controller('smallCatCtrl', function($scope, $http){
    $scope.getSmall = function(){
        $http.get('http://localhost:3000/getSmall').success(function(res) {
            $scope.cats = res;
        });
    };
    $scope.getSmall();
});

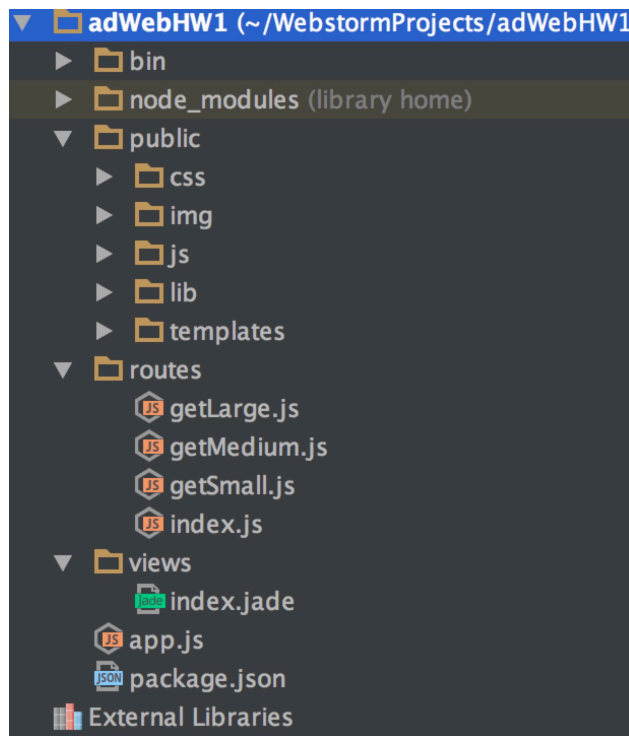
```

分类页面的子页面通过发送请求获取数据（public/js/controllers.js）

2.2. NodeJS 的 Express 框架

Middleware 是 Express.js 非常重要的特性。Express 通过 `app.use` 注册 middleware，middlewares 相当于 request 的 handlers，在处理 request 时，顺序执行每一个 handler（function），Routing 基于 Middleware 的特性匹配不同的 request 到不同的 handler 上。

在本项目中，public 中存储的是前端使用的一些 js 文件，css 文件和几个基本页面；routes 中存储的是根据 app.js 的定义和匹配，不同请求对应的 handler；view 中存储的是该项目的主要页面，也是整个项目的容器，用于展示不同页面的内容。



项目文件

正如前文所说，app.js 定义了不同请求对应的 handler，下图展示的是其中的一个简单示例，返回一个 json 数组给前端。

```
var express = require('express');
var bigcats = [
  {
    "name": "布偶猫",
    "fur": "长毛",
    "weight": "4.5-9KG",
    "color": "大多数单色被毛，玳瑁色花纹和斑纹；总是有重点色、双色和棒球手套状的被毛",
    "character": "布偶猫喜欢人类陪伴，乐于同儿童玩耍，通常对其他宠物友好，但不太好动；",
  },
  {
    "name": "英国短毛猫",
    "fur": "短毛",
    "weight": "4-8KG",
    "color": "单色、重点色、双色、烟色、斑纹、毛尖色、玳瑁色",
    "character": "英国短毛猫大胆好奇，但非常温柔，适应能力也很强，不会因为环境的改变而",
  }
];
var router = express.Router();

router.get('/', function(req, res, next) {
  res.send(bigcats);
});

module.exports = router;
```

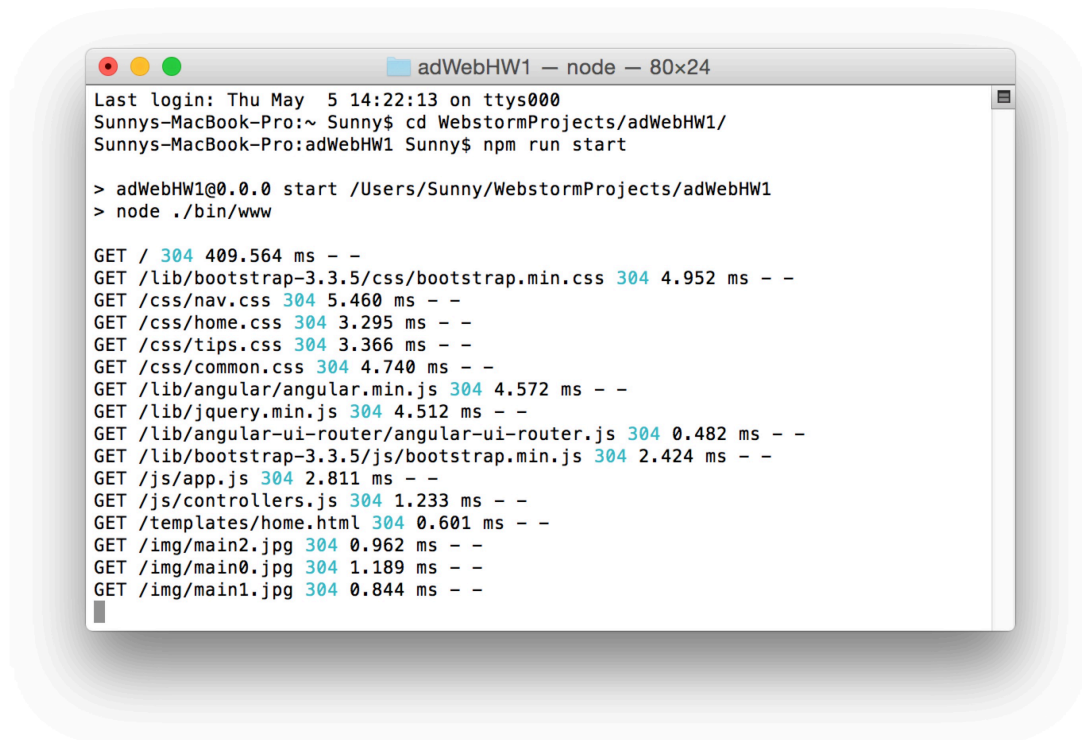
getLarge.js 通过 json 数组返回大型猫的一些信息

2.3. 说明

更多代码细节请直接查看相应文件。由于对 Express 框架了解有限，因此在实现后台时缺乏设计性，文中内容多为个人理解，因此可能与准确的概念出现偏差，特此说明。

3. 运行方法

Terminal 进入项目目录下，输入指令：`npm run start` 运行程序，在浏览器中输入 `localhost:3000` 即可打开页面。



```
adWebHW1 — node — 80x24
Last login: Thu May  5 14:22:13 on ttys000
Sunnys-MacBook-Pro:~ Sunny$ cd WebstormProjects/adWebHW1/
Sunnys-MacBook-Pro:adWebHW1 Sunny$ npm run start

> adWebHW1@0.0.0 start /Users/Sunny/WebstormProjects/adWebHW1
> node ./bin/www

GET / 304 409.564 ms - -
GET /lib/bootstrap-3.3.5/css/bootstrap.min.css 304 4.952 ms - -
GET /css/nav.css 304 5.460 ms - -
GET /css/home.css 304 3.295 ms - -
GET /css/tips.css 304 3.366 ms - -
GET /css/common.css 304 4.740 ms - -
GET /lib/angular/angular.min.js 304 4.572 ms - -
GET /lib/jquery.min.js 304 4.512 ms - -
GET /lib/angular-ui-router/angular-ui-router.js 304 0.482 ms - -
GET /lib/bootstrap-3.3.5/js/bootstrap.min.js 304 2.424 ms - -
GET /js/app.js 304 2.811 ms - -
GET /js/controllers.js 304 1.233 ms - -
GET /templates/home.html 304 0.601 ms - -
GET /img/main2.jpg 304 0.962 ms - -
GET /img/main0.jpg 304 1.189 ms - -
GET /img/main1.jpg 304 0.844 ms - -
```