Easydian Common APIs

目录

[1. schema: 2](#_Toc356656691)

[1.1 Shops: 2](#_Toc356656692)

[1.2 Userinfo: 2](#_Toc356656693)

[1.3 Comment 3](#_Toc356656694)

[1.4 News 3](#_Toc356656695)

[2. Common APIs 3](#_Toc356656696)

[3. API Details 6](#_Toc356656697)

[3.1 shops API 6](#_Toc356656698)

[3.1.1 GET /shops 6](#_Toc356656699)

[3.1.2 GET /shop/:id 7](#_Toc356656700)

[3.1.3 PUT /visit/:id 7](#_Toc356656701)

[3.1.5 POST /shop 8](#_Toc356656702)

[3.1.6 DELETE /shop/:id 8](#_Toc356656703)

[3.1.7 GET /validate\_ field 8](#_Toc356656704)

[3.1.8 PUT /goodbad/:id 8](#_Toc356656705)

[3.2 User APIs 9](#_Toc356656706)

[3.2.1 /new\_user 9](#_Toc356656707)

[3.2.2 GET /user/:id 9](#_Toc356656708)

[3.2.3 PUT /login/:id 9](#_Toc356656709)

[3.2.4 PUT /logout/:id 10](#_Toc356656710)

[3.2.5 POST /register 10](#_Toc356656711)

[3.3 Comment APIs 10](#_Toc356656712)

[3.3.1 GET /comment/:id 10](#_Toc356656713)

[3.3.2 POST /comment/:id 11](#_Toc356656714)

[3.3.3 DELETE /comment/:id 11](#_Toc356656715)

[3.3.4 DELETE /comments/:id 11](#_Toc356656716)

[3.4 News APIs 12](#_Toc356656717)

[3.4.1 GET /news/:id 12](#_Toc356656718)

[3.4.2 DELETE /news/:id 12](#_Toc356656719)

[4. Design 13](#_Toc356656720)

# 1. schema:

## 1.1 Shops:

{

shopname : {type:String, default:'fullname'},

shopalias : {type:Array, default: ['short name']},

shoptype : {type:String, default:'Dining'},

shopvisit : {type:Number, default:1000},

shoppriority : {type: Number, default:1000},

shopwebsite: {type:String, default: 'fullurl'},

shopphone: {type:Array, default: ['010-22222222']},

shoponbusiness: {type:Boolean, default: true},

shoponadv: {type:Boolean, default: true},

shopweekstats:[],

weekday: {type:Array, default:[0,0,0,0,0,0,0]},#visit

weekdaygood: {type:Array, default:[0,0,0,0,0,0,0]},

weekdaybad: {type:Array, default:[0,0,0,0,0,0,0]},

shopgoodt: {type:Number, default:0},

shopbadt: {type:Number, default:0},

shopdaystats:{type:Array,default:[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]},

shopmonthstats: {type:Array,default:[0,0,0,0,0,0,0,0,0,0,0,0]}, #visit

shoplogo: {type:String, default: './images/default.jpg'},

shopcover:{type:Array, default: ['beijing','shanghai']},

shopaccount: {type:Number, default:0},

shopcreatetime: {type:Date, default: Date.now},

shopcommentsnum: {type:Number, default:0}

}

## 1.2 Userinfo:

**Use mongoose format directly.**

@addr\_info = new Schema({

address: {type:String,required:true,unique:true}, # add ",unique:true,trim:true" in

gps:{x:0.00,y:0.00},

usetime: {type:Date, default:Date.now}

})

@phone\_info = new Schema({

phone: {type:String,unique:true}

usetime:{type:Date, default: Date.now}

});

@user\_schema = new Schema({

username: {type:String, required: true}, # add ",unique:true,trim:true" in

useralias: {type:String, default: "Mr Lazy"},

useraddress: [@addr\_info],

userpassword: {type:String, default:"md5"}, #havn't decided how to auth

usertargetaddress: [@addr\_info],

userphone: [@phone\_info], #phone num to be unique

logintime: [{login:Date, logduration:Number}],

belogin: {type:Boolean, default: true}

})

## 1.3 Comment

由于只允许注册用户添加comment， 因此comment里添加一条username

{

shopid : {type:String, default:'shopid'},

username: {type:String, default:'name'},

comment: {type:String, default:'context'},

createtime: {type:Date, default: Date.now}

}

## 1.4 News

**Node.js will not take responsibility of news, crawler based on python will insert news to mongodb directly. Easygo 负责抓取所有的新闻信息。**

{

shopid : {type:String, default:'shopid'},

newsabstract: {type:String, default:'abstract'},

newsurl: {type:String, default:'full\_url'},

newtitle:{type:String, default:'abstract'}

createtime:{type:Date, default: Date.now}

}

# Common APIs

更新controller API：

把原来统一的API 分离为 base 在类别上的API， 不再使用category作为参数来区分

**餐饮API：**

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Method | Parameters | note |
| /canyin/shops | get | start, limit, fields | 获取指定数目的商店  尽量精简fields |
| /canyin/shop/:id | get | fields | 获取一个商店的详细信息 |
| /canyin/shop/:id/visit | put |  | 访问量+1 |
| /canyin/shop | post | Instance of shop | 新增一个商店 |
| /canyin/shop/:id | put | instance of shop | 修改商店 |
| /canyin/shop/:id | delete |  | 删除一个商店 |
| /canyin/shop/:id/bad | put |  | 坏评+1 |
| /canyin/shop/:id/good | put |  | 好评+1 |
| /canyin/shop/:id/comments | get | start, limit  start=0,limit =0 及：全部,（不建议使用） | 获取一个商店的评价  :id 为shop 的id |
| /canyin/shop/:id/comment | post | Username, comment | 插入一条评论,用户必须登录才能发表评论  :id 为shop 的id |
| /canyin/shop/:id/comment | put | Username, comment | 用户更新评论  :id 为comment 的id |
| /canyin/shop/:id/comment | delete | username | :id 为comment 的id |
| /canyin/shop/:id/news | get | Start, limit  start=0,limit =0 及：全部,（不建议使用） | 获取一个商店的新闻  :id 商店的名字 |

**搬家API:**

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Method | Parameters | note |
| /banjia/shops | get | start, limit, fields | 获取指定数目的商店  尽量精简fields |
| /banjia/shop/:id | get | fields | 获取一个商店的详细信息 |
| / banjia/shop/:id/visit | put |  | 访问量+1 |
| / banjia/shop | post | Instance of shop | 新增一个商店 |
| / banjia/shop/:id | put | instance of shop | 修改商店 |
| / banjia/shop/:id | delete |  | 删除一个商店 |
| / banjia/shop/:id/bad | put |  | 坏评+1 |
| / banjia/shop/:id/good | put |  | 好评+1 |
| /banjia/shop/:id/comments | get | start, limit  start=0,limit =0 及：全部,（不建议使用） | 获取一个商店的评价  :id 为shop 的id |
| / banjia/shop/:id/comment | post | Username, comment | 插入一条评论,用户必须登录才能发表评论  :id 为shop 的id |
| / banjia/shop/:id/comment | put | Username, comment | 用户更新评论  :id 为comment 的id |
| / banjia/shop/:id/comment | delete | username | :id 为comment 的id |
| / banjia/shop/:id/news | get | Start, limit  start=0,limit =0 及：全部,（不建议使用） | 获取一个商店的新闻  :name 商店的名字 |

**装修API:**

|  |  |  |  |
| --- | --- | --- | --- |
| URL | Method | Parameters | note |
| /zxiu/shops | get | start, limit, fields | 获取指定数目的商店  尽量精简fields |
| / zxiu /shop/:id | get | fields | 获取一个商店的详细信息 |
| / zxiu /shop/visit/:id | put |  | 访问量+1 |
| / zxiu /shop | post | Instance of shop | 新增一个商店 |
| / zxiu /shop/:id | put | instance of shop | 修改商店 |
| / zxiu /shop/:id | delete |  | 删除一个商店 |
| / zxiu /shop /:id/bad | put |  | 坏评+1 |
| / zxiu /shop /:id/good | put |  | 好评+1 |
| / zxiu /shop /:id/comments | get | start, limit  start=0,limit =0 及：全部,（不建议使用） | 获取一个商店的评价  :id 为shop 的id |
| / zxiu /shop /:id/comment | post | Username, comment | 插入一条评论,用户必须登录才能发表评论  :id 为shop 的id |
| / zxiu /shop /:id/comment | put | Username, comment | 用户更新评论  :id 为comment 的id |
| / zxiu /shop /:id/comment | delete | username | :id 为comment 的id |
| / zxiu /shop /:id/news | get | Start, limit  start=0,limit =0 及：全部,（不建议使用） | 获取一个商店的新闻  :id 为shop 的id |

|  |  |  |  |
| --- | --- | --- | --- |
| For User | | 2 | |
| **URL** | **Method** | **parameters** | **note** |
| /user/register | post | userinfo | 用户注册 |
| /user/:id | put | userinfo | 更新用户 |
| /user/login/:id | put | fields | 用户登录 |
| /user/logout/:id | put |  | 用户退出 |
| /user/:id | delete |  | 删除用户  真的要用吗? |
| /validate\_field | get | Fieldpath, value | 之后需要根据实际需求来实现，在前段可以验证的，尽量放在前段 |

# 3. API Details

## 3.1 shops API

内容太多了，暂时不更新这一部分。 直接交流解释

### 3.1.1 GET /shops

Get the shops registered in easydian

Priority: must

Request Parameters:

Category: specify the category of the shops

Start(optional): start index (sorted by priority), 0 base

Limit(optional): limit the number of items

Request body: N/A

Response Status: 200,400, 500

Response Body: An array of shops, JSON format

{

[{

\_id: xxxx,

shopname: xxxx,

shoplogo: xxxx,

shopwebsite: xxxx,

shopbadt: xxxx,

shopgoodt: xxxx

}],

[]

…

}

### 3.1.2 GET /shop/:id

GET the shop detail

Prority: must

Request Parameters:

Category: specify the category of the shops

Fields: the fields needs to be return, 0 don’t return any fields

Comments: 1: return comments, 0:don’t return comment, sorted by create date

News: 1: return news, 0: don’t return news, sorted by create date

Start: start index of comments/news

Limit: the num of comments/news

Request body: N/A

Response Status: 200, 400, 500

Response Body:

{

Fields:{

Field1: xxxx,

Field2: xxxx,

Field3: xxxx

},

News: [

{News1},

{News2}

],

Comments: [

{Comment1},

{Comment1},

{Comment1}

]

}

### 3.1.3 PUT /visit/:id

Update the visit num of shop

Priority: should

Request Parameters:

Category: specify the category of the shops

Request body: N/A

Response Status: 200, 400, 500

Response body: {visit:value}

### 3.1.5 POST /shop

Insert a new shop to db

Priority: must

Request Parameters:

Category: specify the category of the shops

Request body:

Instance:A json configuration for a shop

Response Status: 200, 400, 500

Response body: only info failed

### 3.1.6 DELETE /shop/:id

Update value of a field

Priority: should

Request Parameters:

Request body:

Response Status: 200, 400, 500

Response body: only info field

### 3.1.7 GET /validate\_ field

Validates the given field’s value

Priority: should

Request Parameters:

table : shop / user

field: the name of the filed

value: the value of the field

Request body: N/A

Response Status: 200, 400, 500

Response body: only info field

### 3.1.8 PUT /goodbad/:id

Update the value of goodbad comment

Priority: must

Request Parameters:

Category: specify the category of the shops

Type: good/bad

Request body: N/A

Response Status: 200, 404, 500

Response body:

{bgt:num}

## 3.2 User APIs

### 3.2.1 /new\_user

Insert a new user to db

Priority: should

Request Parameters: N/A

Request body: A json file of all the user’s information(I’ll prepare it)

Response Status: 200, 400, 500

Response body:only info field

### 3.2.2 GET /user/:id

Get the user’s detail information

Priority: should

Request Parameters:

Fields: the fields that needs to be returned

Request body: N/A

Response Status: 200, 400, 500

Response body: A json file contains user’s information

### 3.2.3 PUT /login/:id

Send user login info to server

Priority: should

Request Parameters: N/A

Request body: a json file of user login info

Response Status: 200, 404, 500

Response body: A json file contains basic user info, / error info

### 3.2.4 PUT /logout/:id

Send user logout info to server

Priority: should

Request Parameters: N/A

Request body: N/A

Response Status: 200, 404, 500

Response body: A json file contains basic user info, / error info

### 3.2.5 POST /register

Insert a user info to db

Priority: should

Request Parameters: N/A

Request body: A json file of user’ info

Response Status: 200, 404, 500

Response body: A json file contains basic user info, / error info

### 3.3 Comment APIs

:id used here should be the ID of a shop

### 3.3.1 GET /comment/:id

Priority: should

Request Parameters:

Start: start index

Limit: num of comments

Level: 0-5, 5 is the best

Category: …

Request body: N/A

Response Status: 200, 400, 500

Response body:

[

{

Comment: xxxxxx

},

{… },

…

]

### 3.3.2 POST /comment/:id

Insert a user info to db

Priority: should

Request Parameters: N/A

Request body: A json file of comment

Response Status: 200, 400, 500

Response body: {\_id:xxxx}

### 3.3.3 DELETE /comment/:id

This is id is the id of a comment

Priority: should

Request Parameters: N/A

Request body: N/A

Response Status: 200, 400, 500

Response body: only info field

### 3.3.4 DELETE /comments/:id

Delete a rank of comments

Priority: should

Request Parameters:

Start: start index

Limit: num of comments

Level: 0-5, 5 is the best, -1 means any Level

Category: …

Request body: N/A

Response Status: 200, 400, 500

Response body: only info field

### 3.4 News APIs

### 3.4.1 GET /news/:id

Priority: should

Request Parameters:

Start: start index

Limit: num of comments

Category: …

Request body: N/A

Response Status: 200, 400, 500

Response body:

[

{

news: xxxxxx

},

{… },

…

]

### 3.4.2 DELETE /news/:id

Priority: should

Request Parameters:

Start: start index

Limit: num of comments

Request body: N/A

Response Status: 200, 400, 500

Response body: only info field

# 4. Design

Node.js

mongodb

redis

View

ejs, jquery

controller

model

Crawler, java