ebookshelf

作者

张子薇 2016011276 钮泽平 2015010467

发布链接

主机ip:123.206.179.98, 端口8000, 日常运行中

- firefox可以通过确认调用摄像头，chrome需要https协议，目前 ebookshelf.cn域名未进行备案，不知道哪天备案通过了就可以访问.....

Github Pages:

Ebookshelf

chrome也可以查看实时扫描功能，但未链接我们的服务器，不能爬取书籍信息。

选题背景

个人的纸质书籍可能太多太乱太杂，希望有一个高效的方法能对其进行统计管理。

功能列表

前端页面

页面设计

导航栏

PC端导航栏采用水平布局，当前选中的页面选项下方会有淡蓝色的下划线，可点击跳转，登录成功后，首页导航栏最右侧会出现用户头像的小图标，点击可跳转到用户已扫过的书目。

移动端导航栏采用下拉菜单的设计，未点击时显示折叠图标，点击后菜单展开。

首页

PC端首页左侧仿照GitHub首页，介绍了网站的功能特点。右侧有注册框和登录框，会向服务器发送相应的用户名和密码。完成登录后会有提示，导航栏出现小头像图标。

移动端功能特点介绍转为竖直排列，注册登陆窗口转移到页面底部。

实时扫描

PC端采用三栏布局的格式，左侧为控制条，包含控制拍照的按钮，右侧为爬取结果的展示页面。中间为摄像头显示界面，点击拍照后拍取画面会显示在摄像画面的下方，若一维码解析不成功会弹出提示。

移动端为保证较好的浏览体验，去掉了拍取画面的显示，控制条转移到摄像画面上方，爬取结果展示画面转移到摄像画面下方。

上传图片

页面布局与实时扫描一致，上传图片后会显示在中间一栏，图片中被认定存在一维码但无法解析的区域会被绿框标出。成功解析的区域会被蓝框标出并用红线标出具体解析的位置。

用户书目

由于时间原因只做了展示界面但未链接数据库，目前显示的书目是以json格式写死在文件内的。

> 使用的开源框架：Bootstrap

> 参考的网站：GitHub首页

前端功能实现

回到顶部

在页面下拉一段距离后在右下角出现，回到top后消失。

一维码扫描

src/javascript/scan.js,src/javascript/upload.js 分别具有handler类用于配置并解析一维码，主要调用开源代码库quaggaJS。目前设置只解析标准13位EAN码，其余格式的一维码无法解析。实现了在前端直接解析一维码的功能。扫描完毕后只向后端传递ISBN码。

> 使用的开源库：quaggaJS

数据传递

> 使用的API：Fetch

结果展示

src/javascript/scan.js,src/javascript/upload.js 利用handleData函数处理服务器传回来的数据并展示给用户，利用React框架更加方便的管理。由于推荐书目的图片大小不一且清晰度较差，利用Bootstrap进行了分栏并且动态的适配移动端。

> 使用的开源框架：Bootstrap,React

后端服务

Python爬虫

利用Python实现的简易爬虫，能通过ISBN码爬取对应书籍的信息。

> 使用的库：Beautiful Soup,urllib2

服务器程序

使用nodejs后端框架koa2进行服务端开发，编写路由解析，动态网页请求响应。

子进程中调用python程序进行信息爬取，包装成json返回客户端。

> 使用中间件：
>
> - koa-bodyparser 用于预处理post请求;
> - koa-router实现请求和处理方法的关联;

> 使用包：
>
> - mime用于获取请求静态资源类型;
> - numjucks用于给koa的conntext请求配置render方法以返回动态渲染的页面;

数据库连接

使用mongoose连接数据库，实现了一个简单的用户信息维护，响应前端对服务器的注册登陆请求。

> 使用了：开源库mongoose. 文档型数据库：mongdb

服务端部署

数据库和服务器程序已经部署在腾讯云主机上，使用nodejs进程管理工具pm2实现其日常运行。

本地测试，安装部署方法：

- 首先安装mongodb，监听默认端口27017并运行
- 安装npm包管理工具，后面的操作在8.x版本下运行通过
- 在根目录下使用npm install自动化安装依赖包
- 运行npm run compile将网页js文件进行babel转换
- 启动node ./app.js 运行服务
- 本地localhost:8000即可访问

遇到的困难

太多了……

- quaggaJS提供的实时识别，摄像头传回的数据波动较大，误识别概率高。

  解决：Web实时识别部分增加按钮捕获当前图像，绘制到canvas中以供查看，相应数据传入QuaggaJS中再进行识别，实现在前端较为稳定的识别

- 爬虫返回的数据中相关图书的图片大小不一，排版困难

  解决：Media query处理了部分情况，并利用Bootstrap栅格系统，对图片进行分栏处理，保证了图片的显示效果，并能较好的适配移动端

- 对于前端开发和各种框架一无所知……

- 前后端交互的各种bug，中间件的编写和前端响应都需要js的异步语法，对js异步语法和中间件工作原理知之甚少给调试server增加了很大困难。不过好在最后server终于相对稳定了

- 之前从未接触过的后端koa框架，mongodb数据库使用，将server应用部署到远程服务器

- babel和npm的设置与使用也耗费了我们不小时间与精力

- https问题，以及域名申请后还需要极为繁杂的备案……

  解决：各种学……

心得感言

- 体验了架设网站完整流程，对网站架构有了一定的认识
- 成功搭建了网站，后期可以进行更多拓展
- 锻炼了快速学习并应用到项目中的能力
- 体会了前端工程化工具带来的便利

从入门到最后完成大作业，感谢助教大大和各位老师对我们的指导和帮助O(∩_∩)O~~