**软件设计文档**

**版本号：1.3**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **修改内容** | **修改日期** | **修改人** |
| **1.0** | **编写文档模板，初步编写技术选型、架构设计** | **2016.7.2** | **何翔** |
| **1.1** | **进一步编写技术选型、架构设计** | **2016.7.2** | **苏逸民** |
| **1.2** | **初步编写服务器设计内容** | **2016.7.3** | **何翔** |
| **1.3** | **修改服务器部分** | **2016.7.3** | **何翔** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Table 1版本管理

目录

[Table 1版本管理 3](#_Toc455265280)

[1.技术选型 4](#_Toc455265281)

[2.架构设计 5](#_Toc455265282)

[3.模块划分 6](#_Toc455265283)

[3.1.前端 6](#_Toc455265284)

[3.2.服务器 6](#_Toc455265285)

[3.2.1.Handler 6](#_Toc455265286)

[3.2.1.Method 7](#_Toc455265287)

[3.3数据库 8](#_Toc455265288)

[3.4豆瓣爬虫 8](#_Toc455265289)

[3.5图书馆爬虫 8](#_Toc455265290)

# 1.技术选型

系统前端采用html、css、javascript以及bootstrap框架等，服务器采用python的tornado框架，数据库使用mysql。这些都是组员熟悉的技术。

爬虫使用python，并利用beautifulsoup解析html。基于html的解析，健壮性更好。

推荐算法使用FP-Growth。这是基于关联规则的推荐。由于爬虫只获取了豆列（类似购物单），不适合使用协同过滤等方法。

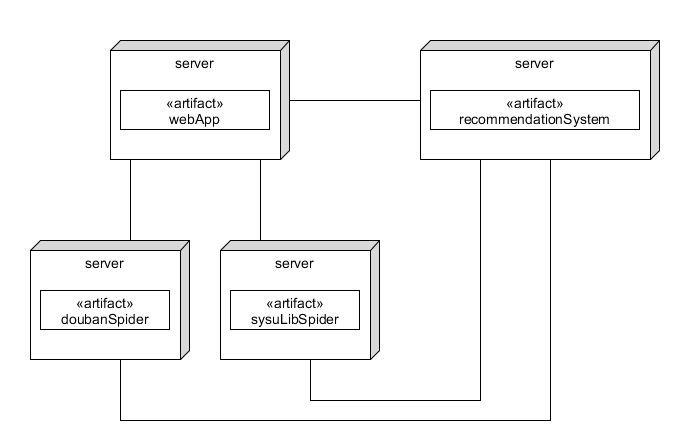
# 2.架构设计

系统由4部分组成，分别为web应用、豆瓣爬虫、图书馆爬虫、推荐算法组成。

爬虫为推荐系统提供训练数据，为web应用提供图书基本信息。

推荐系统为web应用提供推荐书籍列表。

Web应用提供访问服务。



# 3.模块划分

## 3.1.前端

（例如：根据对用例图的分析得出前端5个页面XXX…………………………）

## 3.2.服务器

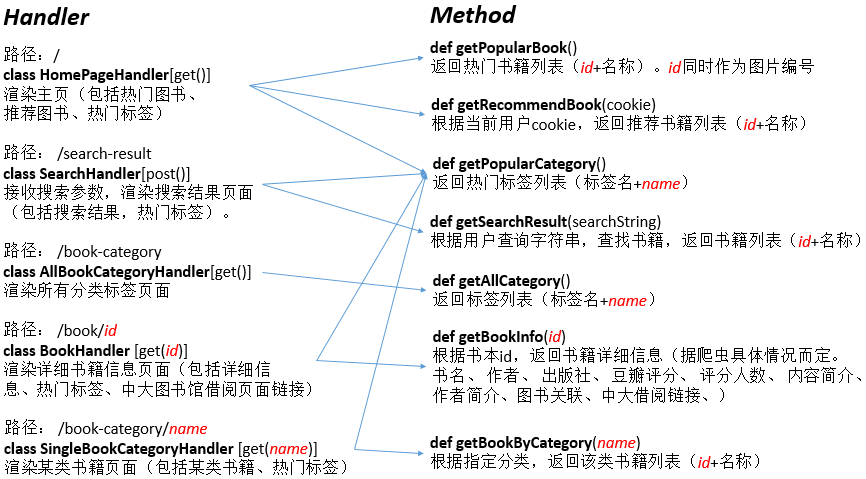


Figure 服务器结构

服务器利用tornado框架搭建，并根据上述前端页面的设计，将服务器结构划分为五个主要Handler：HomePageHandler、SearchHandler、AllBookCategoryHandler、BookHandler、SingleBookCategoryHandler。每个Handler对应于一个Web页面，并根据其渲染的页面所需要的数据，调用相应Method获取。每个Method利用数据库API（设计详见”3.3数据库”章节）从数据库中获取相应数据。

### 3.2.1.Handler

所有Handler均需要判断发起请求的用户是否持有cookie，若cookie为空，则根据当前时间戳分配一个安全cookie。

* **HomePageHandler**

路径：/

get()方法用于渲染主页，渲染内容为热门图书（图书名称、图片、链接，下同）、推荐图书、热门标签（标签内容、链接，下同）。在该方法中，通过调用getPopularBook()、getRecommendBook()、getPopularCategory()三个方法来获取所需数据。

渲染内容：popularBook, recommendBook, popularCategory。

popularBook：[[id, bookName],]

recommendBook：[[id, bookName],]

popularCategory：[[name, content],]

post()方法需考虑容错处理。

* **SearchHandler**

路径：/search-result

post()方法用于处理用户提交的搜索表单，并渲染搜索结果页面，渲染内容为热门图书、热门标签。在该方法中，通过调用getSearchResult(searchString)、getPopularCategory()两个方法来获取所需数据。

渲染内容：searchBook, popularCategory。

searchBook：[[id, bookName],]

popularCategory：[[name, content],]

get()方法需考虑容错处理。

* **AllBookCategoryHandler**

路径： /book-category

get()方法渲染所有分类标签页面，渲染内容为标签。在该方法中，通过调用getAllCategory()方法来获取所需数据。

渲染内容：allCategory。

allCategory：[[name, content],]

post()方法需考虑容错处理。

* **BookHandler**

路径： /book/id

get()方法用于渲染详细书籍信息页面，渲染内容为详细信息：书名、作者、出版社、豆瓣评分、评分人数、内容简介、作者简介、图书关联、中大图书馆借阅页面链接（该信息由爬虫获取后存入数据库，设计详见”3.4豆瓣爬虫”、“3.5图书馆爬虫”章节）、热门标签，在该方法中，首先获取get请求url中的书籍id，再通过调用getBookInfo(id)方法来获取所需数据。

渲染内容：bookInfo, popularCategory。

bookInfo：[书名,作者,出版社,豆瓣分,评分人数,内容简介,作者简介,中大借阅链接,[[id, bookName], ]]（[[id, bookName],]为图书关联列表）

popularCategory：[[name, content],]

post()方法需考虑容错处理。

* **SingleBookCategoryHandler**

路径： /book-category/name

get()方法渲染某一类书籍页面，渲染内容为某类标签、热门标签。在该方法中，首先获取get请求url中的分类标签，再通过调用getBookByCategory(name)方法来获取所需数据。

渲染内容：categoryBook, popularCategory。

categoryBook：[[id, bookName],]

popularCategory：[[name, content],]

post()方法需考虑容错处理。

### 3.2.1.Method

* **getPopularBook()**

函数通过调用数据库API并进行相关操作，找出n个热门书籍（点击次数最多），返回一个热门书籍列表[[id, bookName], ]。

id：书籍编号，构成前端中每个书籍的链接（/book/id）

bookName：书籍的名称

* **getRecommendBook(cookie)**

根据用户cookie值，调用数据库API进行相关操作，根据用户浏览记录利用推荐算法，找出推荐书籍，并返回一个推荐书籍列表[[id，bookName],]。

id：同上

bookName：同上

* **getPopularCategory()**

调用数据库API进行相关操作，返回一个热门（点击次数最多）分类标签列表[[name, content],]。

name：分类标签英文，构成前端中每个标签的链接（/book-category/name）

content：分类标签中文

* **getSearchResult(searchString)**

根据用户提交的表单内容，调用数据库API进行书籍的模糊查询。返回一个书籍列表[[id, bookName], ]。

* **getAllCategory()**

调用数据库API返回一个标签列表[[name, content],]。

* **getBookInfo(id)**

根据书籍id值，调用数据库API，返回书本详细信息列表[书名,作者,出版社,豆瓣分,评分人数,内容简介,作者简介,中大借阅链接,[[id, bookName], ] ]。

[[id, bookName],]：图书关联列表

* **getBookByCategory(name)**

根据分类标签name值，调用数据库API，返回一个该类书籍列表[[id, bookName], ]。

## 3.3数据库

## 3.4豆瓣爬虫

## 3.5图书馆爬虫