

**《Django应用开发》期末**

**报告**

题 目： 图书管理系统

学 院： 计算机与人工智能

专 业： 计算机科学与技术

班 级： 18计本1

学 号： \*\*\*\*\*\*\*\*\*\*\*

姓 名： 宁争胜

指导老师： \*\*\*\*\*\*

完成日期： 2020年5月28日

**目录**

**[第1章 项目简介](#_Toc14318_WPSOffice_Level1)** **[3](#_Toc14318_WPSOffice_Level1)**

[1.1 前言](#_Toc25906_WPSOffice_Level2) [3](#_Toc25906_WPSOffice_Level2)

[1.2 开发环境](#_Toc3789_WPSOffice_Level2) [3](#_Toc3789_WPSOffice_Level2)

[1.3 开发与部署](#_Toc11468_WPSOffice_Level2) [3](#_Toc11468_WPSOffice_Level2)

**[第2章 需求分析](#_Toc25906_WPSOffice_Level1)** **[1](#_Toc25906_WPSOffice_Level1)**

[2.1 业务内容](#_Toc22349_WPSOffice_Level2) [1](#_Toc22349_WPSOffice_Level2)

[2.2 功能模块](#_Toc2942_WPSOffice_Level2) [1](#_Toc2942_WPSOffice_Level2)

**[第3章 数据库设计与实现](#_Toc3789_WPSOffice_Level1)** **[2](#_Toc3789_WPSOffice_Level1)**

[3.1概念模型（ER图）](#_Toc20543_WPSOffice_Level2) [2](#_Toc20543_WPSOffice_Level2)

[3.2数据库逻辑结构设计](#_Toc10418_WPSOffice_Level2) [2](#_Toc10418_WPSOffice_Level2)

[3.3数据库实现](#_Toc2851_WPSOffice_Level2) [2](#_Toc2851_WPSOffice_Level2)

**[第4章 关键技术介绍](#_Toc11468_WPSOffice_Level1)** **[4](#_Toc11468_WPSOffice_Level1)**

[4.1 项目框架](#_Toc16544_WPSOffice_Level2) [4](#_Toc16544_WPSOffice_Level2)

[4.2 通用头文件header.html的设置](#_Toc26918_WPSOffice_Level2) [4](#_Toc26918_WPSOffice_Level2)

[4.3 主界面的图片滚动效果](#_Toc13902_WPSOffice_Level2) [5](#_Toc13902_WPSOffice_Level2)

[4.4 图书查询的get和post请求](#_Toc21913_WPSOffice_Level2) [6](#_Toc21913_WPSOffice_Level2)

[4.5 Ajax方式更新图书](#_Toc4771_WPSOffice_Level2) [8](#_Toc4771_WPSOffice_Level2)

[4.6 管理员导出EXCEL表格](#_Toc15671_WPSOffice_Level2) [10](#_Toc15671_WPSOffice_Level2)

[4.7 部署Django项目到Linux服务器](#_Toc16659_WPSOffice_Level2) [11](#_Toc16659_WPSOffice_Level2)

[4.8 添加图书是对所填信息进行验证](#_Toc28837_WPSOffice_Level2) [13](#_Toc28837_WPSOffice_Level2)

**[第5章 项目说明书](#_Toc22349_WPSOffice_Level1)** **[13](#_Toc22349_WPSOffice_Level1)**

[5.1 图书管理系统的使用](#_Toc7855_WPSOffice_Level2) [13](#_Toc7855_WPSOffice_Level2)

[5.2 图书管理系统功能介绍](#_Toc21445_WPSOffice_Level2) [14](#_Toc21445_WPSOffice_Level2)

**[第6章 项目总结](#_Toc2942_WPSOffice_Level1)** **[16](#_Toc2942_WPSOffice_Level1)**

**[第7章 参考文献](#_Toc20543_WPSOffice_Level1)** **[16](#_Toc20543_WPSOffice_Level1)**

# 项目简介

1.1 前言

进入21世纪以来，信息技术从根本上推动了图书馆的飞速发展，计算机和计算机管理系统已成为图书馆进行图书管理的主要设备和系统。虽然目前很多大型的图书馆已经有一整套比较完善的管理系统，但是在一些中小型的图书馆中，大部分工作仍需手工完成，工作起来效率比较低，不便于动态、及时地调整图书结构。为了更好地适应当前图书馆的管理需求，解决手工管理中存在的弊端，越来越多的中小型图书馆正在逐步向计算机信息化管理转变。

1.2 开发环境

本机编程软件：JetBrains PyCharm 2018.1.1 x64

阿里云服务器：CentonOS

python：3.6

django：2.1.4

mysql：5.7

web服务器：nginx

wsgi服务器：uwsgi

python连接mysql库：pymysql

1.3 开发与部署

* 本地编程开发Django图书管理系统；
* 阿里云服务器的环境准备、安装环境，数据库迁移；
* 部署方式：项目配置django+nginx+uwsgi；
* 最后本地测试；

# 需求分析

* 1. 业务内容

主页部分：上面会自动展示最新的的通知或者书籍推荐；

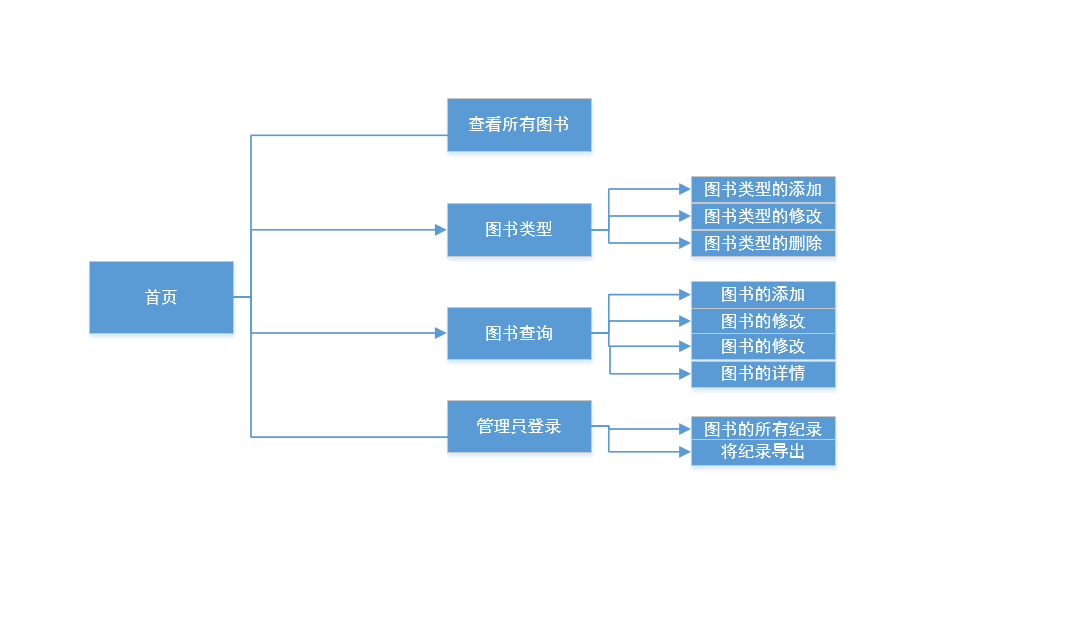
图书类型部分：用户可以对图书的类型进行添加、修改、删除，图书类型在后面的图书中需要用到。图书类型：图书类别，类别名称，可借阅天数 。

查看所有图书部分：用户可以在此界面上查看数据库中所有的图书。

图书部分：用户可以对图书进行添加、修改、删除。图书：图书条形码，图书名称，图书所在类别，图书价格，库存，出版日期，出版社，图书图片，图书简介，图书文件。

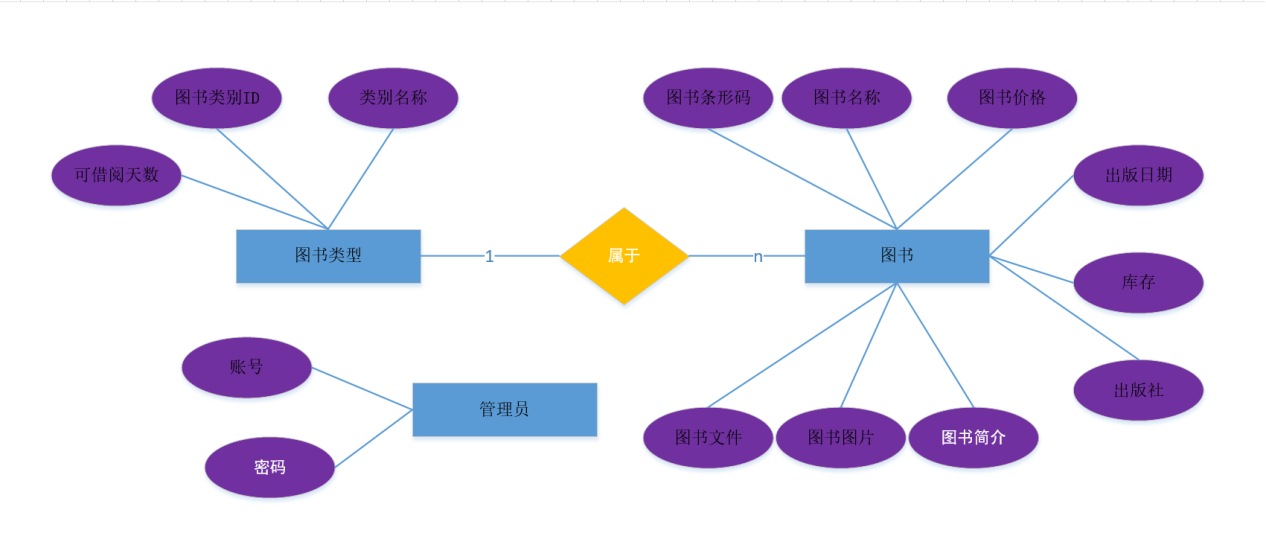
管理员部分：管理员可以查看数据库中所有信息，并可以将自己需要的信息以表格的形式导到自己的设备上。

2.2 功能模块



# 数据库设计与实现

3.1概念模型（ER图）



3.2数据库逻辑结构设计

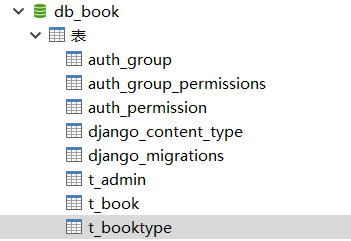
1. 图书类型：（图书类别，类别名称，可借阅天数）
2. 图书：（图书条形码，图书名称，图书所在类别，图书价格，库存，出版日期，出版社，

图书图片，图书简介，图书文件）

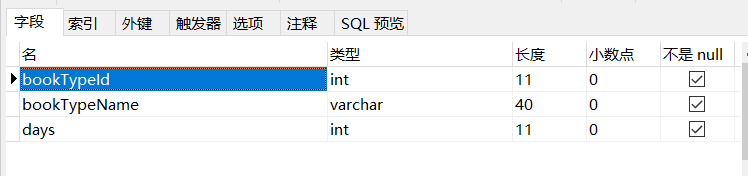
1. 管理员：（账号，密码）

经检查，以上经汇总ER图转换得到的数据表不存在部份依赖，也不存在函数依赖，符合第三正规化。由此可以开始进行数据库的实现过程。

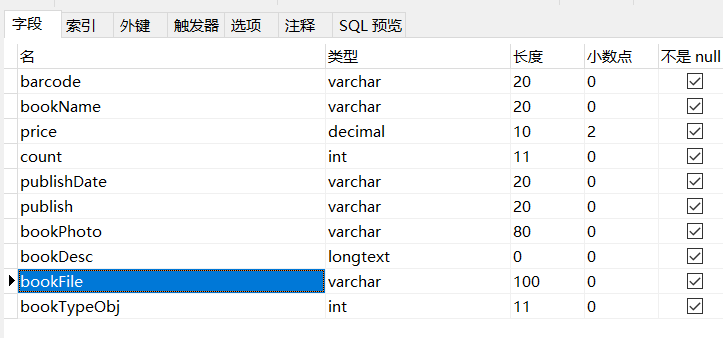
3.3数据库实现



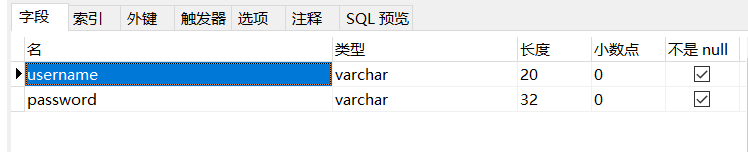
**图书类型表：**



**图书表：**



**管理员表：**



# 关键技术介绍

* 1. 项目框架

项目一共创建三个App，统一放在apps包中进行管理，在apps中创建了一个baseView.py文件用来供三个app的公共函数调用。创建了一个media文件夹，这个文件夹用来保存项目上次的图片、书籍文件，以及导出文件。创建了一个static文件夹，这里面主要是一些css和js的包。templates里面是项目所有的的html文件。

路由的设置，在项目的总urls.py文件中设置路由到各个app模块，然后在每个app下面的urls.py文件中，设置具体的html定位。

urlpatterns **=** [  
 **path**(**'admin/'**, admin.site.urls),  
 **url**(**r'^media/(?P<path>.\*)$'**, serve, {**'document\_root':**settings.MEDIA\_ROOT}),*#这部分很重要* **url**(**r'^BookType/'**, **include**(**'apps.BookType.urls'**, namespace**='BookType'**)), *# 图书类别模块* **url**(**r'^Book/'**, **include**(**'apps.Book.urls'**, namespace**='Book'**)), *# 图书模块* **url**(**r'^'**, **include**(**"apps.Index.urls"**, namespace**="Index"**)), *# 首页模块*]

* 1. 通用头文件header.html的设置

在网页的整个展示过程中头文件是一直需要展示出来的，所有我就将这个标题栏部分单独做成了header.html文件，这样可以极大的减少代码的冗余，也方便编写程序。

<!--导航-->  
<**div** class=**"navbar-collapse collapse"**>  
 <**ul** class=**"nav navbar-nav navbar-left"**>  
 <**li**><**a** href=**"**{% **url** 'Index:index' %}**"** style=**"color**:grey**"**>**首页**</**a**></**li**>  
 <**li**><**a** href=**"**{% **url** 'Book:frontAll' %}**"** style=**"color**:grey**"**>**所有图书**</**a**></**li**>  
 <**li**><**a** href=**"**{% **url** 'BookType:frontList' %}**"** style=**"color**:grey**"**>**图书类型**</**a**></**li**>  
 <**li**><**a** href=**"**{% **url** 'Book:frontList' %}**"** style=**"color**:grey**"**>**图书查询**</**a**></**li**>  
 <**li**><**a** href=**"**{% **url** 'Index:login' %}**"** style=**"color**:grey**"**>**登录**</**a**></**li**>  
 </**ul**>  
</**div**>  
<!--导航-->

**导航栏：**

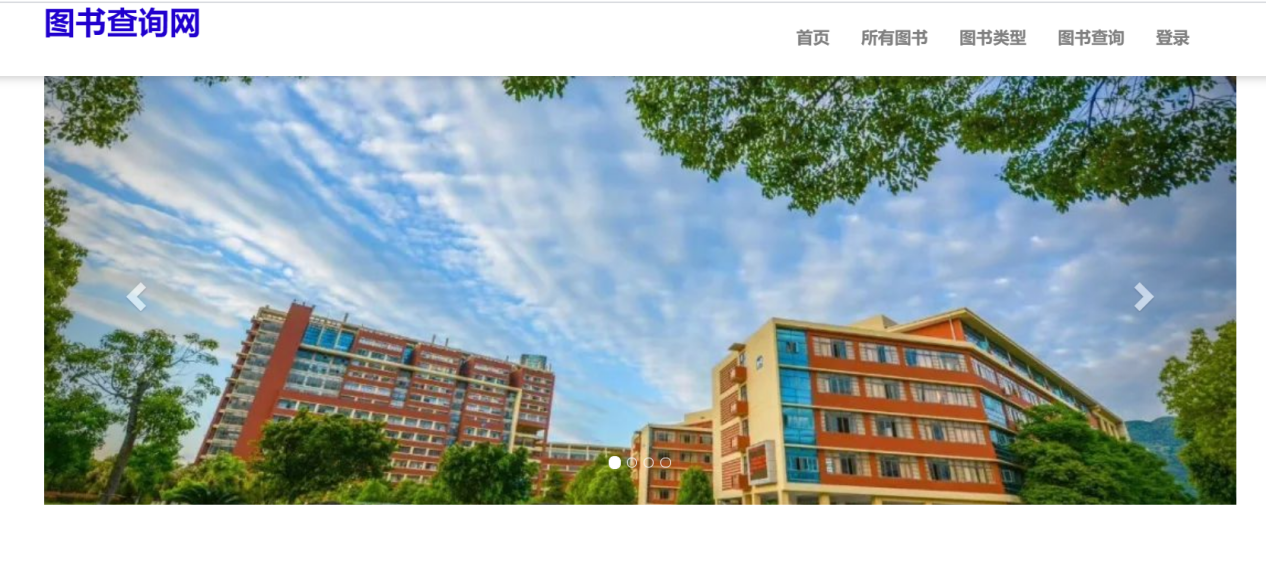


* 1. 主界面的图片滚动效果

图片的滚动需要css样式以及js代码一起控制产生效果，下面是js代码介绍。

参考博客：<https://www.cnblogs.com/zmshare/p/6091102.html>

|  |
| --- |
| **1 $(function() {  6 function resize() {  7 // 获取屏幕宽度  8 var windowWidth = $(window).width();  9 // 判断屏幕属于大还是小 10 var isSmallScreen = windowWidth < 768; 11 // 根据大小为界面上的每一张轮播图设置背景 13 $('#main\_ad > .carousel-inner > .item').each(function(i, item) { 14 var $item = $(item); 15 // var imgSrc = $item.data(isSmallScreen ? 'image-xs' : 'image-lg'); 16 var imgSrc = 17 isSmallScreen ? $item.data('image-xs') : $item.data('image-lg'); 18 // 设置背景图片 19 $item.css('backgroundImage', 'url("' + imgSrc + '")'); 20 // 21 // 因为我们需要小图时 尺寸等比例变化，所以小图时我们使用img方式 22 if (isSmallScreen) { 23 $item.html('**<**img** src=**"' + imgSrc + '"** alt=**""** />**'); 24 } else { 25 $item.empty(); 26 } 27 }); 28 } 30 // // 让window对象立即触发一下resize，初始化div的背景图 31 // $(window).trigger('resize'); 34 $(window).on('resize', resize).trigger('resize'); 35 });** |



* 1. 图书查询的get和post请求

图书查询界面在开始加载的时候的get请求，这个时候需要界面上查询控件里的内容全部为空，这种情况下，需要查询所有图书信息显示到界面上。由于post请求也需要调用查询的功能，其区别就是查询条件不一样而已，这样就可以写一个handle函数供两种不同请求来调用。

对于post请求，首先需要对用户输入内容进行不同的验证，以确保查询语句的正确性。

最后将从数据库查询的内容加工组合成一个字典，通过render函数返回给**Book/book\_frontquery\_result.html界面显示出来。**

**class FrontListView**(BaseView)**:** *# 前台图书查询列表* **def get**(self, *request*)**:  
 return** self.**handle**(*request*)  
  
 **def post**(self, *request*)**:  
 return** self.**handle**(*request*)  
  
 **def handle**(self,*request*)**:** self.**getCurrentPage**(*request*) *# 获取当前要显示第几页  
 # 下面获取查询参数* barcode **=** self.**getStrParam**(*request*, **'barcode'**)  
 bookName **=** self.**getStrParam**(*request*, **'bookName'**)  
 publishDate **=** self.**getStrParam**(*request*, **'publishDate'**)  
 bookTypeObj\_bookTypeId **=** self.**getIntParam**(*request*, **'bookTypeObj.bookTypeId'**)  
  
 print(barcode,bookName,publishDate,bookTypeObj\_bookTypeId)  
 *# 然后条件组合查询过滤* book **=** Book.objects.**all**()  
 **if** barcode **!= '':** book **=** book.**filter**(barcode\_\_contains**=**barcode)  
 **if** bookName **!= '':** book **=** book.**filter**(bookName\_\_contains**=**bookName)  
 **if** publishDate **!= '':** book **=** book.**filter**(publishDate\_\_contains**=**publishDate)  
 **if** bookTypeObj\_bookTypeId **!= '0':** book **=** book.**filter**(bookTypeObj**=**bookTypeObj\_bookTypeId)  
 *# 对查询结果利用Paginator进行分页* self.paginator **= Paginator**(book, self.pageSize)  
 *# 计算总的页码数，要显示的页码列表，总记录等* self.**calculatePages**()  
 *# 获取第page页的Page实例对象* books\_page **=** self.paginator.**page**(self.currentPage)  
  
 *# 获取所有图书类型* bookTypes **=** BookType.objects.**all**()  
 *# 构造模板需要的参数* context **=** {  
 **'bookTypes':** bookTypes,  
 **'books\_page':** books\_page,  
 **'barcode':** barcode,  
 **'bookName':** bookName,  
 **'publishDate':** publishDate,  
 **'bookTypeObj\_bookTypeId':** int(bookTypeObj\_bookTypeId),  
 **'currentPage':** self.currentPage,  
 **'totalPage':** self.totalPage,  
 **'recordNumber':** self.recordNumber,  
 **'startIndex':** self.startIndex,  
 **'pageList':** self.pageList,  
 }  
 *# 渲染模板界面* **return render**(*request*, **'Book/book\_frontquery\_result.html'**, context)

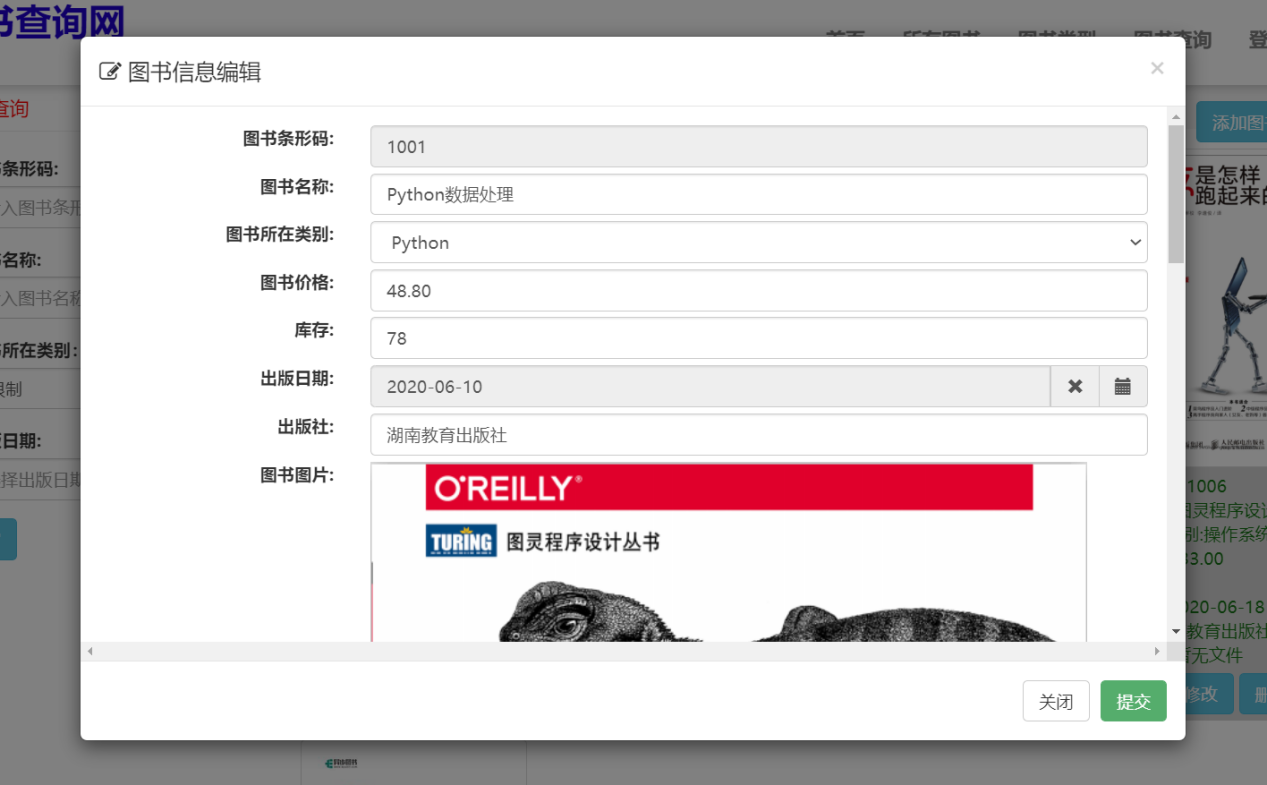


* 1. Ajax方式更新图书

当用户点击修改按钮时，在界面上会弹出一个修改图书信息的模态对话框，下面代码则是实例化模态框以及弹出的JS代码。

首先需要实例化图书描述编辑器当点击修改按钮时，将发起ajax请求，这是将会弹出修改的对话框，这时需要初始化对话框里面的内容，初始化：就是将所点击修改的图书的所有信息获得后，再放到对话框对应的控件中，为后面的修改做准备。

*//实例化图书描述编辑器* tinyMCE.**init**({  
 selector**:** "#book\_bookDesc\_edit",  
 theme**:** 'advanced',  
 language**:** "zh",  
 strict\_loading\_mode**:** 1,  
 });  
  
/\*弹出修改图书界面并初始化数据\*/  
**function** *bookEdit*(*barcode*) {  
 $.ajax({  
 url **:** '/Book/update/' **+** *barcode*,  
 type **:** "get",  
 dataType**:** "json",  
 **success : function** (*book*, *response*, *status*) {  
 **if** (*book*) {  
 $("#book\_barcode\_edit").**val**(*book*.barcode);  
 $("#book\_bookName\_edit").**val**(*book*.bookName);  
 $.ajax({  
 url**:** "{% **url** 'BookType:listAll' %}",  
 type**:** "get",  
 **success: function**(*bookTypes*,*response*,*status*) {  
 $("#book\_bookTypeObj\_bookTypeId\_edit").**empty**();  
 **var** html**=**"";  
 $(*bookTypes*).**each**(**function**(*i*,*bookType*){  
 html **+=** "<option value='" **+** *bookType*.bookTypeId **+** "'>" **+** *bookType*.bookTypeName **+** "</option>";  
 });  
 $("#book\_bookTypeObj\_bookTypeId\_edit").**html**(html);  
 $("#book\_bookTypeObj\_bookTypeId\_edit").**val**(*book*.bookTypeObjPri);  
 }  
 });  
 $("#book\_price\_edit").**val**(*book*.price);  
 $("#book\_count\_edit").**val**(*book*.count);  
 $("#book\_publishDate\_edit").**val**(*book*.publishDate);  
 $("#book\_publish\_edit").**val**(*book*.publish);  
 $("#book\_bookPhotoImg").**attr**("src", *book*.bookPhoto);  
 tinyMCE.editors['book\_bookDesc\_edit'].**setContent**(*book*.bookDesc);  
 $("#book\_bookFileA").**text**(*decodeURIComponent*(*book*.bookFile));  
 $("#book\_bookFileA").**attr**("href", *book*.bookFile);  
 $('#bookEditDialog').modal('show');  
 } **else** {  
 *alert*("获取信息失败！");  
 }  
 }  
 });  
}



* 1. 管理员导出EXCEL表格

**管理员在登录成功后，可以导出数据库中的图书数据，当点击导出按钮时，OutToExcelView将会处理导出的数据处理。首先对数据进行检验，然后条件组合查询过滤**

**class OutToExcelView**(BaseView)**:** *# 导出图书信息到excel并下载* **def get**(self,*request*)**:** *# 收集查询参数* barcode **=** self.**getStrParam**(*request*, **'barcode'**)  
 bookName **=** self.**getStrParam**(*request*, **'bookName'**)  
 publishDate **=** self.**getStrParam**(*request*, **'publishDate'**)  
 bookTypeObj\_bookTypeId **=** self.**getIntParam**(*request*, **'bookTypeObj.bookTypeId'**)  
 *# 然后条件组合查询过滤* books **=** Book.objects.**all**()  
 **if** barcode **!= '':** books **=** books.**filter**(barcode\_\_contains**=**barcode)  
 **if** bookName **!= '':** books **=** books.**filter**(bookName\_\_contains**=**bookName)  
 **if** publishDate **!= '':** books **=** books.**filter**(publishDate\_\_contains**=**publishDate)  
 **if** bookTypeObj\_bookTypeId **!= '0':** books **=** books.**filter**(bookTypeObj**=**bookTypeObj\_bookTypeId)  
 *#将查询结果集转换成列表* bookList **=** []  
 **for** book **in** books**:** book **=** book.**getJsonObj**()  
 bookList.**append**(book)  
 *# 利用pandas实现数据的导出功能* pf **=** pd.**DataFrame**(bookList)  
 *# 设置要导入到excel的列* columns\_map **=** {  
 **'barcode': '图书条形码'**,  
 **'bookName': '图书名称'**,  
 **'bookTypeObj': '图书类别'**,  
 **'price': '图书价格'**,  
 **'count': '图书数量'**,  
 **'publishDate': '出版日期'**,  
 **'publish': '出版社'**,  
 }  
 pf **=** pf[columns\_map.**keys**()]  
 pf.**rename**(columns**=**columns\_map, inplace**=True**)  
 *# 将空的单元格替换为空字符* pf.**fillna**(**''**, inplace**=True**)  
 *#设定文件名和导出路径* filename **= 'books.xlsx'** *# 这个路径可以在settings中设置也可以直接手动输入* root\_path **=** settings.MEDIA\_ROOT **+ '/output/'** file\_path **=** os.path.**join**(root\_path, filename)  
 pf.**to\_excel**(file\_path, encoding**='utf-8'**, index**=False**)  
 *# 将生成的excel文件输出到网页下载* file **=** open(file\_path, **'rb'**)  
 response **= FileResponse**(file)  
 response[**'Content-Type'**] **= 'application/octet-stream'** response[**'Content-Disposition'**] **= 'attachment;filename="books.xlsx"'  
 return** response



* 1. 部署Django项目到Linux服务器
     1. Linux 安装Python3.6

sudo add-apt-repository ppa:jonathonf/python-3.6#这句话的意思是获取最新的个人软件包档案源，将其添加至当前apt库中，并自动导入公钥。

sudo apt-get update# update命令，会访问源列表里的每个网址，并读取软件列表，然后保存在本地电脑。我们在新立得软件包管理器里看到的软件列表，都是通过update命令更新的。

sudo apt-get install python3.6

* + 1. 安装Django

pip3 install django

* + 1. 安装Nginx

sudo apt-get update

sudo apt-get install nginx

* + 1. 安装Uwsgi

apt-get install python3.6-dev

pip install uwsgi

* + 1. Uwsgi测试py文件

uwsgi --http :8001 --wsgi-file test.py

* + 1. 上传Django项目到阿里云服务器
    2. 配置uwsgi

[uwsgi]

# Django-related settings

# 指定项目执行的端口号

socket = :8011

# Django项目目录

chdir = /home/my\_web

# wsgi.py,直接写个项目名.wsgi就可以了

module = my\_web.wsgi

# process-related settings

# master

# 表示以主进程模式运行

master = true

# 进程数

processes = 4

# ... with appropriate permissions - may be needed

# chmod-socket = 664

# clear environment on exit

vacuum = true

#保存日志文件，这个路径就是刚刚新建的myuwsgi.log文件的路径

daemonize = /home/my\_web/myuwsgi.log

# 表示不记录正常信息，只记录错误信息，否则你的日志可能很快就爆满了

disable-logging = true

#在项目生成两个文件uwsgi.status和uwsgi.pid 方便查看pid和运行状态

stats=/home/my\_web/uwsgi.status

pidfile=/home/my\_web/uwsgi.pid

* + 1. 配置Nginx

server {

listen 80;

server\_name www.xxx.com

charset UTF-8;

access\_log /var/log/nginx/myweb\_access.log;

error\_log /var/log/nginx/myweb\_error.log;

client\_max\_body\_size 75M;

location / {

include uwsgi\_params;

uwsgi\_pass 127.0.0.1:8011;

uwsgi\_read\_timeout 2;

}

}

* 1. 添加图书是对所填信息进行验证



# 项目说明书

* 1. 图书管理系统的使用

可以通过http://112.74.109.77:8011/访问图书管理系统，本网站支持电脑端，同时也支持手机端的访问。

下面看一下电脑端和手机端访问成功后的效果图：

**电脑端：**



手机端效果；

* 1. 图书管理系统功能介绍

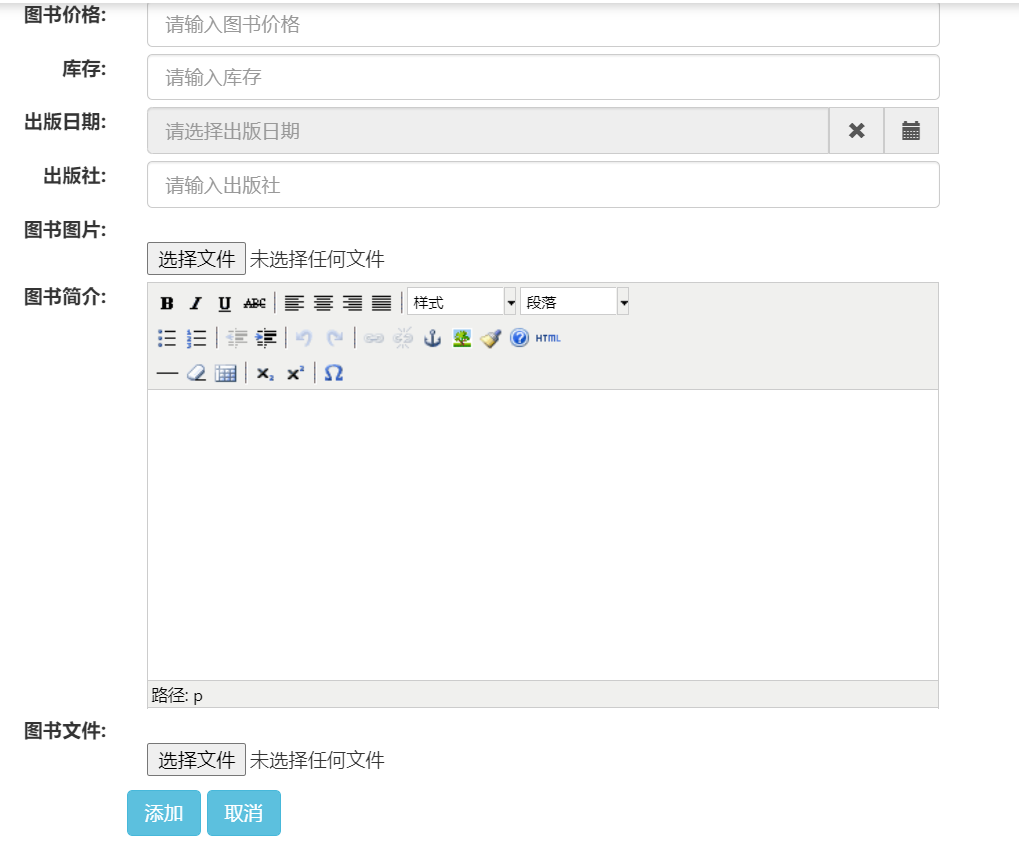
1.对所有的图书类型的增删查



2.对所有的图书的增删查



**添加：**



**修改：**



3.用户登录后台



# 项目总结

我觉得这门课最后考察通过大作业的形式真的特别的好，为什么这么说呢，感觉这几天学到的东西特别的多。我平常看书写作业最晚也就到12点的样子，就上床睡觉了，但是课程设计的编程就写的特别的投入，感觉写出来很有成就感，我记得我前几天晚上改bug改到了凌晨3点，虽然最后还是没有改出来，在深夜改代码的过程中我精神特别的好，那个时候想起了计算机网络老师说的一句话：你们打游戏到深夜很开心，当你们改代码bug到深夜时，你们会更开心。我觉得老师说的很对。

从这个大作业开始构建框架到最后的完成，花了好久天的时间，在整个开发的过程中遇到了很多的问题难题，然后自己就疯狂百度，看了大量的博文。在网上查找自己想要的信息还是有点难受的，因为网上会有大量的信息，查出来的信息中有一部分会有很多的问题，可能本来就是错误的或者这个解决方案并不适合自己当下的环境。从问题的出现，到自己网上查找解决方案，这可能要花费很长的时间，但是我个人觉得这是很值得的，因为这一过程锻炼了自己解决问题的能力，而不是当初出现了问题就避开这个难题，如果还是想当初那样遇到问题就逃避，这样自己是不会有进步的。不管是学习还是生活中，遇到困难，都应该勇敢的去面对。

最后送自己一句话：明天成功的自己会感谢今天努力的自己！

# 参考文献

1. [Bootstrap中文网](http://www.bootcss.com/" \t "https://v3.bootcss.com/css/_blank)：<https://v3.bootcss.com/css/>

2.Font Awesome：<https://fontawesome.dashgame.com/>

3.滚动图片的制作：https://www.cnblogs.com/zmshare/p/6091102.html

4.Linux启动Django : <https://www.cnblogs.com/gaoya666/p/9159774.html>

# 5.pycharm创建djang项linux部署<https://blog.csdn.net/Angry_Mills/article/details/79249645>