暨南大學
JINAN UNIVERSITY

本科实验报告

课程名称：数据库系统原理实验

课程编号：08060309

学生姓名：甄洛生

学号：2018054625

学院：信息科学技术学院

系：计算机科学系

专业：计算机科学与技术

指导教师：黄穗

教师单位：计算机科学系

开课时间：2020~ 2021 学年度第 1 学期

暨南大学教务处

2020 年　月　日

数据库系统原理实验　项目目录

学生姓名：甄洛生　　学号：2018054625

序号	实验项目编号	实验项目名称	*实验项目类型	成绩	指导教师
1	1	期刊管理系统	综合性		黄穗
2					
3					
4					
5					
6					
7					
8					
9					
10					
11					
12					
13					
14					
15					
16					
17					
18					

*实验项目类型：演示性、验证性、综合性、设计性实验。

*此表由学生按顺序填写。

暨南大学本科实验报告专用纸

课程名称 数据库系统原理实验 成绩评定

实验项目名称 期刊管理系统 指导教师 黄穗

实验项目编号 1 实验项目类型 综合性 实验地点

学生姓名 甄洛生 学号 2018054625

学院 信息科学技术 系 计算机系 专业 计算机科学与技术

实验时间 2020 年 11 月 12 日～ 11 月 25 日

一、实验主题

本实验旨在开发出带 GUI 的期刊管理系统，系统记录和检索各种期刊信息，可生成统计报表用于期刊信息的统计。期刊的初始数据集来源于 2008 年中国百种杰出学术期刊，所有数据存放于关系型数据库 MySQL 中。期刊信息可包含：期刊名称,英文刊名,CN,ISSN,主编,主办单位,地址,邮编,电话,传真,电子信箱,出版周期,创刊时间。为了更好地管理数据和查阅数据，使用系统需要账号登陆，账号分为管理员级别和普通用户级别。管理员可对期刊进行任何操作；普通用户只能查阅期刊信息，打印报表等。

二、实验设计

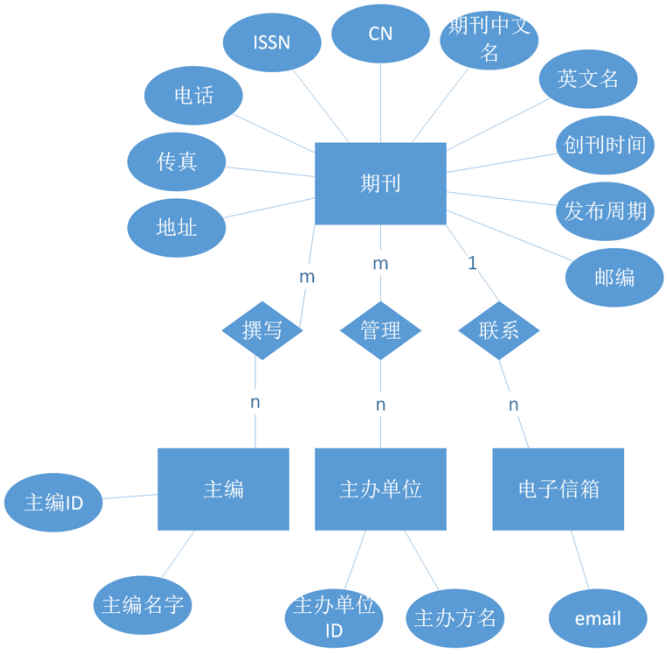
1. 设计思路

A. ER 图设计

浏览 2008 年中国百种杰出学术期刊后，得出以下结论：

1. 电话，传真，电子信箱是可以为空的。
2. 主编，主办单位与期刊是多对多关系。
3. 期刊和电子信箱是一对多关系。
4. 创刊时间大于 1800 年

故 ER 图设计如下：



B. 账户注册逻辑

考虑到功能的完整性与便于管理，只需要提供用户名、邮箱和密码进行注册。用户名和邮箱是唯一的，并且用户的密码以加密的形式存储于数据库中，保证账户安全性。

C. 账户登录逻辑

考虑项目便于使用，登录时使用邮箱、密码验证。若不存在该邮箱，则返回账户不存在消息。若密码错误，则提示密码有误信息。

D. 增添期刊逻辑

设计表单，管理员填写好必要的期刊信息后，点击提交。客户端取得表单数据，检查数据格式是否有误，检查数据是否与现有数据重复，如果一切正常，才插入期刊数据至数据库。

E. 删除期刊逻辑

删除期刊按钮位于期刊详情界面内，流程可通过查询期刊->期刊详情->删除期刊来进行期刊的删除操作。删除前，先判断当前用户是否具备删除权限，若有，则根据期刊信息操作数据库进行删除操作。

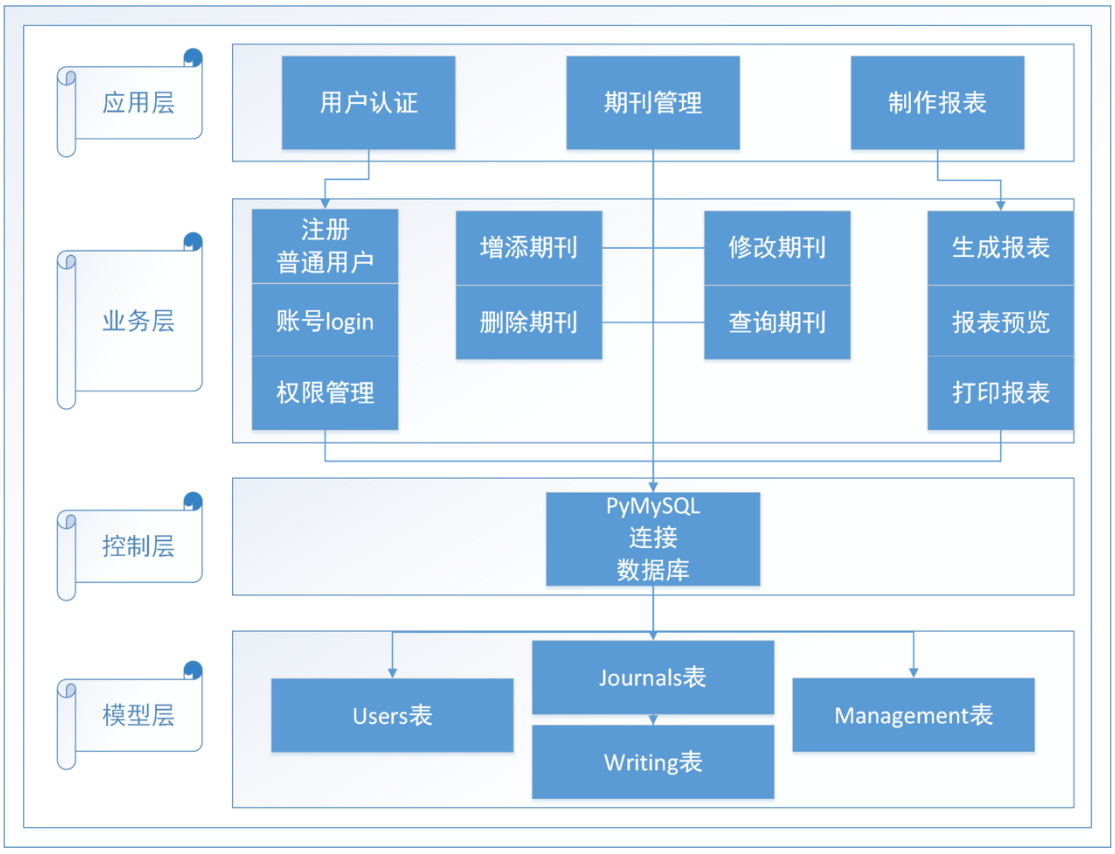
F. 查询期刊逻辑

设计三种不同的查询方式，一个是以中文刊名进行查询，另一个是以 ISSN 号进行查询，最后一个是以 CN 号进行查询。所有的查询都要求模糊匹配。

G. 修改期刊逻辑

修改期刊按钮位于期刊详情界面内，流程可通过查询期刊->期刊详情->修改期刊来进行期刊的修改操作。删除前，先判断当前用户是否具备修改权限，若有，则根据期刊信息生成表单，用户修改表单内容再提交至客户端，客户端会判断数据的正确性和冗余性判断期刊的修改是否成功。

2. 架构图



三、软件实现

项目主要用 Python/Javascript 开发。分别使用了 Flask 和 Electron 两大框架构建服务端与客户端。客户端发送操作请求，服务端相应请求进行操作。

先介绍 Flask 和 Electron 所担任的角色：

1. *Flask*

Flask 是一个用 Python 编写的微 Web 框架。Flask 自动帮我们解析来自客户端的 Http 请求信息，我们只需编写相应 URL 的响应函数即可。Flask 之所以被称为微框架，因为它做的工作也仅限于此。对于 POST 的表单处理，数据库操作，登录认证等功能都需要自己编写，这样的好处就是自由度很高，可以用自己喜欢的方式实现。缺点就是开发周期较长。

2. *Electron*

Electron 让我们可以使用 JavaScript，HTML 和 CSS 构建跨平台的桌面应用程序。大名鼎鼎的 Visual Studio Code 和 Atom 就是基于 Electron 开发的。简单来说，Electron 内置了 Chromium 渲染引擎和 NodeJS 运行时。Chromium 渲染引擎负责渲染接收到的 html 和 css 文件。NodeJS 运行时使得 Javascript 可以操作桌面级交互，比如通知栏提醒，构建原生菜单栏等。

A. 服务端 Server

服务器绑定至本地 IP：127.0.0.1，端口为 5000。服务器持续监听来自客户端的请求，解析客户端请求资源路径，调用对应资源路径相应方法。比如：

```
1.  # 主界面
2.  @main.route('/index')
3.  @login_required
4.  def index():
5.      # 获取数据库所有 journal 数据
6.      journals = Journal.query.all()
7.      username = current_user.username
8.      flash('目前期刊数: {}'.format(len(journals)), 'info')
9.      return render_template('index.html', journals=journals, username=usernam
    e)
```

这段代码指明，服务端解析 http 请求得到资源路径”/index”后，自动调用对应 index()方法，服务器获取期刊表所有数据，并动态生成 html 文件，返回至客户端，Electron 的 Chromium 渲染引擎就会渲染 html 文件进行显示。

B. 客户端界面设计

由于我们使用 HTML+CSS 的技术来设计 GUI，我们的 GUI 设计的自由性比较强，可以根据自己的想法设计。传统的 GUI 设计如 C++的 QT，Java 的 Swing 和 JavaFX，这些都是基于操作系统提供的内置 GUI 组件进行 GUI 设计，因此设计出来的 GUI 比较符合原操作系统的风格。

出于方便和观赏性，我参考了 Bootstrap 官网的几个示例页面，在这些页面的基础上修改得到适合我们项目的 GUI。

C. 客户端 Client

客户端最核心的功能就是制作报表。客户端首先点击“生成报表”，此时客户端就会发送报表 Html 请求，服务器返回报表 Html 至客户端，客户端接收到 Html 的内容后本地转成 PDF。此时 PDF 还未标注页码。于是，使用 reportLab 对生成的 PDF 进行加工，添加上页码，

现在报表制作成功。再在客户端点击“报表预览”即可进行对 PDF 的预览。

D. **数据集录入**

由于老师提供的数据集是以 PDF 的形式记录，并且数据量大，总共 100 个期刊，每个期刊至少都有 10 个属性字段，人工录入非常的耗时耗力。因此我用了 Python 的 tika 模块，将 PDF 中的文字提取出来，现在就可以用 Python 的 re 库分割出每个期刊和期刊的每个属性。

四、软件 demo

本软件基于跨平台开发，Windows/MacOS/Linux 均可使用，但使用前需安装：

A. Python3

B. Node.js

C. MySQL8

D. Java1.8（Python tika 库需要使用 Java）

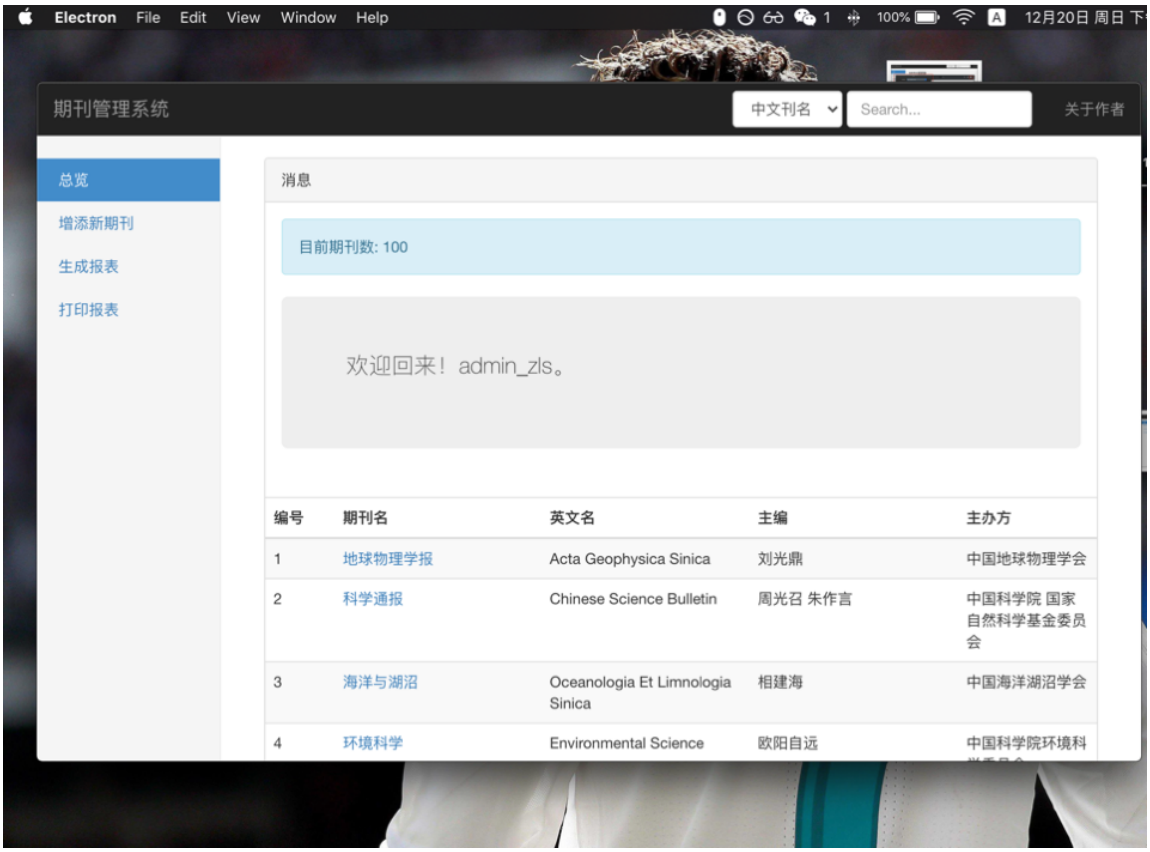
环境配置文档及**源代码**参见：https://github.com/astzls213/database-HW

Demo 环境：（MacOS10.14.5/Python3.7.2/MySQL8.0.17/NodeJS12.18.4）

1. Login 界面

2. Regiter 界面

3. 主界面



4. 期刊详情演示

5. 增添期刊演示

插入期刊信息：

插入结果：

6. 删除期刊演示

删除刚刚的测试期刊：

删除成功：

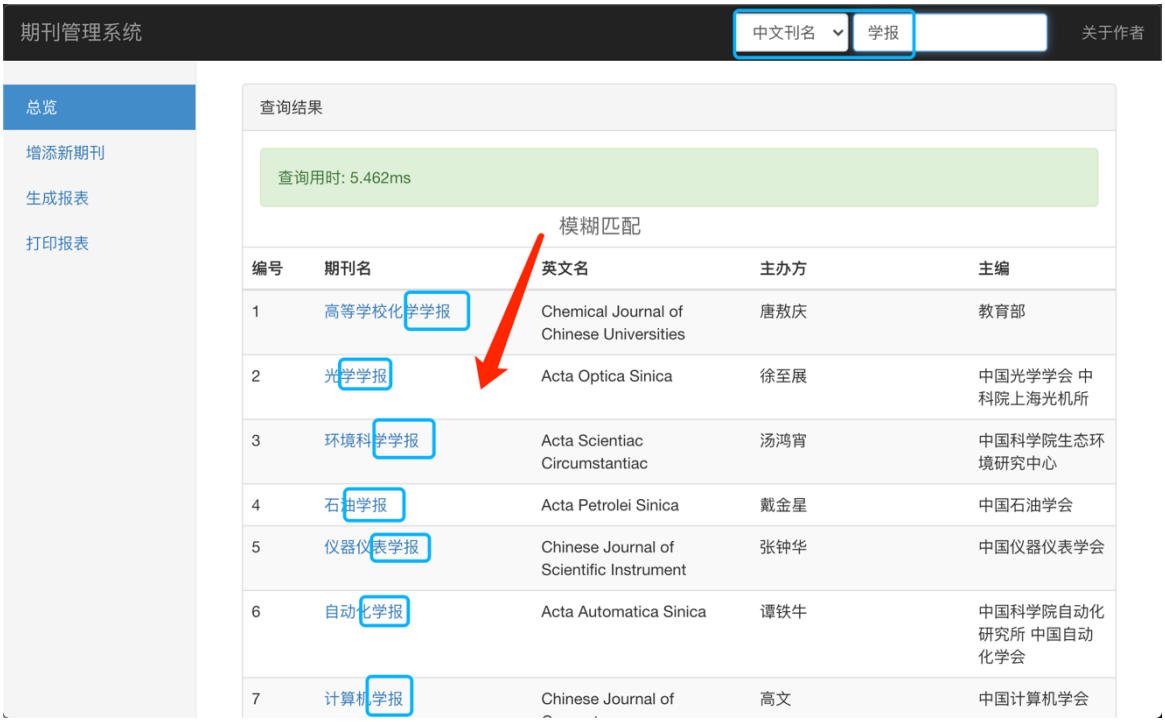
7. 修改期刊信息演示


中文刊名: 地球物理学报
修改为月球物理学报
英文刊名: Acta Geophysica Sinica
ISSN: 0001-5733
修改为1110-5733
CN: 11-2074/P
主编: 刘光鼎
主办单位: 中国地球物理学会
创刊时间: 1948
出版周期(期/年): 6
地址: 北京市9825信箱
邮编: 100101
电邮: actageop@mail.igcas.ac.cn
电话: 010-62007700
传真: 010-62007709
修改期刊信息
删除该期刊


中文刊名: 月球物理学报
英文刊名: Acta Geophysica Sinica
ISSN: 1110-5733
CN: 11-2074/P
主编: 刘光鼎
主办单位: 中国地球物理学会
创刊时间: 1948
出版周期(期/年): 6
地址: 北京市9825信箱
邮编: 100101
电邮: actageop@mail.igcas.ac.cn
电话: 010-62007700
传真: 010-62007709
修改期刊信息
删除该期刊

8. 查询期刊演示

模糊匹配结果：



9. 制作报表演示

五、实验总结

通过本次实验，我最大的收获就是了解到前后端是如何协作处理的。所谓前端，就是指客户端部分的业务逻辑，使用 html/css/javascript 来构建网页。而后端要做的工作就是解析处理来自客户端的请求，比如客户端想增添一个期刊，那么前端就会向后端发送请求信息 D，D 也就是所谓的 HTTP 协议信息。后端通过网络适配器接收到了 D，进行解析。解析得知客户端想要增添期刊，跳转至增添期刊响应函数 F。函数 F 从解析后的 D 获得期刊信息 I，再分析 I 中数据与数据库是否冗余？是否不符合格式要求？若不符合就返回错误信息至客户端。若符合，则注入响应 SQL 语句至数据库执行，增添期刊。数据库 insert 成功返回消息至后端，后端得知增添成功后，再返回消息至客户端。

其次，了解到后端是如何与数据库进行交互的。对于 Java 后端，一般就是使用 JDBC 连接数据库。而对于 Python 后端，一般通过 PyMySQL 模块连接 MySQL 进行操作。并且，操纵数据库不一定非要使用原生 SQL 语句进行编程，还可使用 ORM 对象关系映射数据库的表。当开发项目的时候，如果不使用 ORM，可能会写 SQL 的代码，用来从数据库保存、删除、读取对象信息等等，而这些代码写起来很多重复，并且有 SQL 注入的危险。ORM 解决的主要问题是对象关系的映射。一般情况下，一个持久化类和一个表对应，类的每个实例对应表中的一条记录，类的每个属性对应表的每个字段。使用 ORM 大大提高了开发效率。ORM 提供了对数据库的映射，不用写 SQL 语句编程，就能够像操作对象一样与数据库进行交互。在本实验中，使用的 ORM 框架是 Flask-SQLAlchemy。

暨南大学本科实验报告专用纸(附页)

附录：

主要功能的源代码清单：

1、登录界面：
login.html:

```
{% extends 'auth_base.html' %}

{% block page_content %}
    <div class="container">
        <form class="form-signin" method="POST">
            <h2 class="form-signin-heading">请登录</h2>

            {{ wtf.quick_form(form) }}
        </form>
        <p>
            新用户？
            <a href="{{ url_for('main.register') }}">
                点击这里注册！
            </a>
        </p>

        {% for tag, msg in get_flashed_messages(with_categories=true) %}
        <div class="alert alert-{{ tag }}" role="alert">
            {{ msg }}
        </div>
        {% endfor %}
    </div>
{% endblock %}
```

blueprint.py（部分）：

```
@main.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    # POST
    if form.validate_on_submit():
        # 检查邮箱
        user = User.query.filter_by(email=form.email.data).first()
        # 检查密码
        if user is not None and user.verify_password(form.passwd.data):
```

```
            # 登录成功
            login_user(user)
            return redirect(url_for('main.index'))
        else:
            # 登录失败
            flash('无效的用户名或密码！', 'danger')
            return redirect(url_for('main.login'))
    # GET
    logout_user()   # 先登出用户，无论如何
    return render_template('login.html', form=form)
```

2、更新功能：

blueprint.py（部分）：

```
@main.route('/update/<issn>', methods=['GET','POST'])
@login_required
def update(issn):
    update_form = JournalForm()
    # 检查当前用户是否有权限
    if current_user.type == False:
        flash('此操作无权限对于当前用户。', 'danger')
        return redirect(url_for('main.index'))
    journal = Journal.query.filter_by(issn=issn).first()
    # POST
    if update_form.insert_submit.data and update_form.validate():
        start = datetime.now()
        # 检查 issn/cn 是否重复
        sanity_check = []
        sanity_check.append(Journal.query.filter_by(issn=update_form.issn.data).first())
        sanity_check.append(Journal.query.filter_by(cn=update_form.cn.data).first())
        if sanity_check[0] is not None:
            if sanity_check[0].issn != journal.issn:
                flash('ISSN: {} 已存在数据库中！请仔细检查。'.format(journal.issn), 'danger')
                return redirect(url_for('main.insert'))
        if sanity_check[1] is not None:
            if sanity_check[1].cn != journal.cn:
                flash('CN: {} 已存在数据库中！请仔细检查。'.format(journal.cn), 'danger')
                return redirect(url_for('main.insert'))
        # 替换 '' 为 None
        if update_form.tel.data == '':
            update_form.tel.data = None
```

```
        if update_form.fax.data == '':
            update_form.fax.data = None
        # 先删除之前的信息
        for tmp in journal.emails:
            db.session.delete(tmp)
        for tmp in journal.editors:
            journal.editors.remove(tmp)
        for tmp in journal.organizers:
            journal.organizers.remove(tmp)
        # 先提交删除 否则后面外键不统一
        db.session.commit()
        # 基本字段的修改
        journal.cn_name = update_form.cn_name.data
        journal.en_name = update_form.en_name.data
        journal.issn = update_form.issn.data
        journal.cn = update_form.cn.data
        journal.address = update_form.address.data
        journal.tel = update_form.tel.data
        journal.fax = update_form.fax.data
        journal.public_year = update_form.public_year.data
        journal.public_cycle = update_form.public_cycle.data
        journal.zipcode = update_form.zipcode.data
        # 一对多 多对多关系修改
        emails = update_form.emails.data.split('/')
        editors = update_form.editors.data.split('/')
        organs = update_form.organizers.data.split('/')

        for loop_item in emails:
            if loop_item == '': # 跳过空项
                continue
            if Email.query.filter_by(email=loop_item).first() is None:
                db.session.add(Email(loop_item, journal))
            else:
                flash('此邮箱已经存在，请仔细检查！', 'danger')
                return redirect(url_for('main.insert'))

        for loop_item in editors:
            if loop_item == '': # 跳过空项
                continue
            tmp = Editor.query.filter_by(name=loop_item).first()
            if tmp is None:     # 如果此前没出现过这个主编
                journal.editors.append(Editor(loop_item))
            else:
                tmp.journals.append(journal)
```

```
72.
73.         for loop_item in organs:
74.             if loop_item == '': # 跳过空项
75.                 continue
76.             tmp = Organizer.query.filter_by(name=loop_item).first()
77.             if tmp is None:     # 如果此前没出现过这个主办方
78.                 journal.organizers.append(Organizer(loop_item))
79.             else:
80.                 tmp.journals.append(journal)
81.         # 提交
82.         db.session.commit()
83.         # 返回总览 并给出插入成功提示
84.         end = datetime.now()
85.         msg = '修改用时: %.3fms' % ((end-start).total_seconds()*1000,)
86.         flash('期刊信息修改成功! ' + msg, 'success')
87.         return redirect(url_for('main.index'))
88.     # GET 自动填充原先字段
89.     update_form.cn_name.data = journal.cn_name
90.     update_form.en_name.data = journal.en_name
91.     update_form.issn.data = journal.issn
92.     update_form.cn.data = journal.cn
93.     update_form.public_year.data = journal.public_year
94.     update_form.public_cycle.data = journal.public_cycle
95.     update_form.address.data = journal.address
96.     update_form.zipcode.data = journal.zipcode
97.     update_form.tel.data = journal.tel
98.     update_form.fax.data = journal.fax
99.     # Email/主编/主办单位的填充稍微麻烦些
100.     padding = ''
101.     for tmp in journal.emails:
102.         padding = padding + tmp.email + '/'
103.     update_form.emails.data = padding[:-1] # 丢弃末尾 '/'
104.     padding = ''
105.     for tmp in journal.editors:
106.         padding = padding + tmp.name + '/'
107.     update_form.editors.data = padding[:-1] # 丢弃末尾 '/'
108.     padding = ''
109.     for tmp in journal.organizers:
110.         padding = padding + tmp.name + '/'
111.     update_form.organizers.data = padding[:-1] # 丢弃末尾 '/'
112.     flash('1. *为必填项', 'info')
113.     flash('2. 电子邮箱/主编/主办单位多个时请用 / 隔开','info')
114.     return render_template('jour_form.html', form=update_form)
```

3、查询功能：
blueprint.py（部分）：

```
@main.route('/search', methods=['POST'])
@login_required
def search():
    # 获取 POST 数据
    option = request.form.get('option', None)
    found = request.form.get('search', '')
    # 检查输入栏是否为空
    if found is '':
        flash('输入栏为空!','info')
        return redirect(url_for('main.index'))
    else:
        start = datetime.now()  # 记录开始时间点
        # 根据 Option 查找与 found 模糊匹配的数据库
        if option == 'cn_name':
            found = Journal.cn_name.like('%{}%'.format(found))
            results = Journal.query.filter(found).all()
        elif option == 'issn':
            found = Journal.issn.like('{}%'.format(found))
            results = Journal.query.filter(found).all()
        elif option == 'cn':
            found = Journal.cn.like('{}%'.format(found))
            results = Journal.query.filter(found).all()
    end = datetime.now()  # 记录结束时间点
    msg = '查询用时: %.3fms' % ((end-start).total_seconds()*1000,)
    flash(msg,'success')
    # 渲染结果
    return render_template('sear_result.html', results=results)
```

4、报表功能：
blueprint.py（部分）：

```
# 返回报表 PDF 路由
@main.route('/pdf')
@login_required
def pdf():
    path = 'report.pdf'
    base = os.path.basename(path)

    tmp = "__tmp.pdf"

    batch = 10
    batch = 0
```

```
    output = PdfFileWriter()
    with open(path, 'rb') as f:
        pdf = PdfFileReader(f,strict=False)
        n = pdf.getNumPages()
        if batch == 0:
            batch = -n
        createPagePdf(n,tmp)
        with open(tmp, 'rb') as ftmp:
            numberPdf = PdfFileReader(ftmp)
            for p in range(n):
                if not p%batch and p:
                    newpath = path.replace(base, base[:-4] + '_page_%d'%(p//batch) + path[-4:])
                    with open(newpath, 'wb') as f:
                        output.write(f)
                    output = PdfFileWriter()
                page = pdf.getPage(p)
                numberLayer = numberPdf.getPage(p)

                page.mergePage(numberLayer)
                output.addPage(page)
            if output.getNumPages():
                newpath = path.replace(base, 'final.pdf')
                with open(newpath, 'wb') as f:
                    output.write(f)

        os.remove(tmp)
    return send_file('../final.pdf')

# 打印报表视图
@main.route('/print_report')
@login_required
def print_report():
    return render_template('print_report.html')

# 报表视图 请勿添加 login_required
# 否则 electron 打不开页面
@main.route('/use_for_reporting')
def report_table():
    journals = Journal.query.all()
    today = str(datetime.now()).split()[0]
    return render_template('report.html', journals=journals, current_date=today)
```

renderer.js（部分）：

```
var report_btn = document.getElementById('gen_report');

report_btn.addEventListener('click', () => {
    // 新建一个隐藏的页面
    const pdfWin = new BrowserWindow({
        frame: true,
        show: false,
        webPreferences: {
            nodeIntegration: true,
            nativeWindowOpen: true
        }
    })

    // 前往报表页面
    pdfWin.loadURL('http://localhost:5000/use_for_reporting')

    // 生成 PDF
    pdfWin.webContents.on('did-finish-load', () => {
        // Use default printing options
        pdfWin.webContents.printToPDF({
            printBackground: true
        }).then(data => {
            const pdfPath = path.join('./report.pdf')
            fs.writeFile(pdfPath, data, (error) => {
                if (error) throw error
                console.log(`Wrote PDF successfully to ${pdfPath}`)
            })
            }).catch(error => {
            console.log(`Failed to write PDF to ${pdfPath}: `, error)
        })
    })

    // 提示成功消息
    const notification = {
        title: '消息',
        body: '报表已生成'
    }

    const myNotification = new window.Notification(notification.title, notif
ication)

    myNotification.onclick = () => {
        console.log('OK')
```

```
43.     }
44. })
```