网络编程技术文挡

专业：网络工程

班级：2015211313

姓名：吕滢 2015211514

李子茜 2015211516

（一） 需求分析

1. 在线问答系统的设计与实现

设计实现一个简易的类“知乎”的在线问答系统

2. 基本要求：

有基本的用户管理功能

有问题列表（所有用户可见）

可以查看针对问题的回答（所有用户可见）

注册用户可以发布问题

注册用户可以回答问题

3. 扩展要求（选做其中的几项）：

增加关注功能

增加回复提醒功能

回答问题时可以加入图片

根据问题的讨论人次，形成热点话题列表

给出性能测试结果

（二） 设计

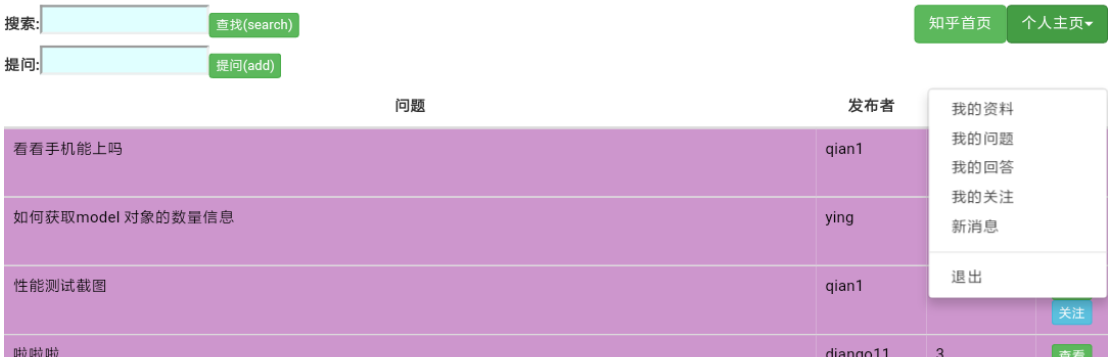
1. 模型

定义了三个模型，截图如下

```
class Question(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=10)#问题标题最长10个字符
    mode =  models.BooleanField(default=False) #表示该问题是否有更新 True还没被查看过 False已经查看过了 增加回答时置为 True，点开问题时
    #content = models.CharField(max_length=200)#问题描述可以长一点
    answerNum = models.IntegerField(default=0) #用于首页显示热度，排序 增加 (question, myquestion) 删除回答 (myquestion, myanswer) 两
    focus = models.IntegerField(default=0) #主页中显示是否关注

class Answer(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)
    user= models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.CharField(max_length=150)
    img = models.ImageField(upload_to='upload', null=False, blank=True, default = os.path.join ( '/','upload','empty.png'),)

class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    focusedquestion = models.ManyToManyField(Question)
    introduction = models.CharField(max_length=150)



        #def save(self, *args ,*kwargs):
    #   super(UserProfile,self).save()
    #    for i in self.foquestionlist:
    #        p,created = Question.objects.get_or_create(name = i)
    #        self.focusedquestion.add(p)
    #    self.foquestionlist[]
```

Class	属性	功能
Question	User	提问的用户，外键
	Title	问题的标题
	Mode	是否被提问者看过，在消息提醒的时候起作用

	AnswerNum	一共有多少个回答，在形成热点话题时有用
	Focus	主页中显示是否被关注，和 js 一起更新页面，现实和已关注或者关注
Answer	Question	是哪个问题的回答，外键
	User	回答者，外键
	Content	回答内容
	Img	问题添加的图片
UserProfile	User	用户外键
	Focusedquestion	关注的问题 many to many 多对多关系
	Introduction	个人简介

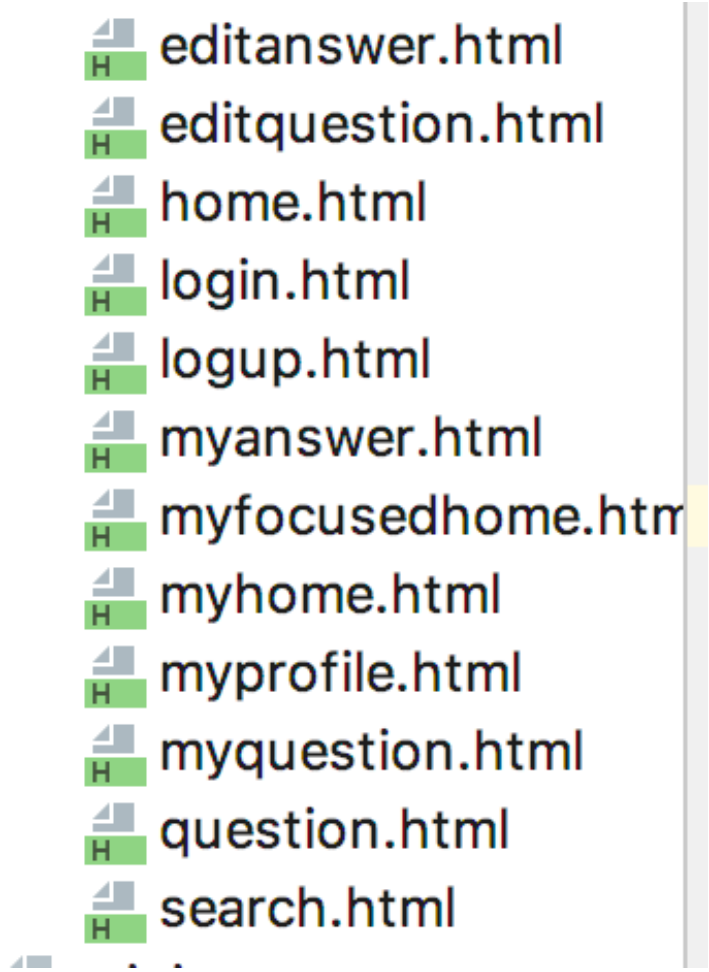


如图，在任何一个页面上都有知乎首页按钮可以回到首页，还有个人主页按钮可以进行相关查询，比如查看个人资料、查看我的问题、我的关注、新消息提醒以及退出按钮。

左上方保证，随时可以搜索问题，随时可以提问问题。

2.html 文件

截图如下



序号	文件名	文件功能	负责人
1	Home	知乎首页	李子茜
2	Login	登录页面	吕滢
3	Logup	注册界面	吕滢
4	Question	问题界面	吕滢
5	Search	搜索界面	李子茜
6	Myhome	个人主页	吕滢
7	myanswer	我的回答	李子茜
8	Myfocusedhome	我的关注	李子茜
9	Myprofile	我的简介	吕滢
10	Editanswer	编辑回答界面	李子茜
	Editquestion	编辑问题界面	李子茜

3.views 函数

序号	函数	功能	负责人
1	logup	注册	吕滢
2	loginView	登陆	吕滢
3	home	知乎首页	李子茜+吕滢
4	search	搜索	李子茜
5	logoutView	注销	吕滢
6	user	用户管理	吕滢
7	myhome	我的首页	李子茜
8	questiondel	删除问题	吕滢
9	myquestion	我的问题	李子茜
10	answeradmindel	删除自己问题的回答	吕滢
11	questionadminedit	编辑自己的问题	李子茜
12	myanswer	我的回答	李子茜
13	answerdel	删除自己的回答	李子茜
14	answeredit	编辑自己的回答	李子茜
15	focusquestion	关注问题	李子茜+吕滢
16	myfocusedhome	我的关注	李子茜
17	focusedquestiondel	取关	吕滢
18	myprofile	我的简介	吕滢
19	editpassword	修改密码	吕滢
20	editprofile	编辑个人简介	吕滢
21	mynewanswer	回答问题	李子茜

（三）实现

模型定义

问题模型

```
class Question(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    title = models.CharField(max_length=10)#问题标题最长 10 个字符
    mode =  models.BooleanField(default=False) #表示该问题是否有更新 True 还没被查看过 False 已经查看过了 增加回答时置为 True，点开问题时（question，myquestion）置为 False
    #content = models.CharField(max_length=200)#问题描述可以长一点
    answerNum = models.IntegerField(default=0) #用于首页显示热度，排序 增加（question，myquestion）删除回答（myquestion，myanswer）时利用当前问题的回答数赋
```

```
值# 0108ying
    focus = models.IntegerField(default=0) #主页中显示是否关注
```

回答模型

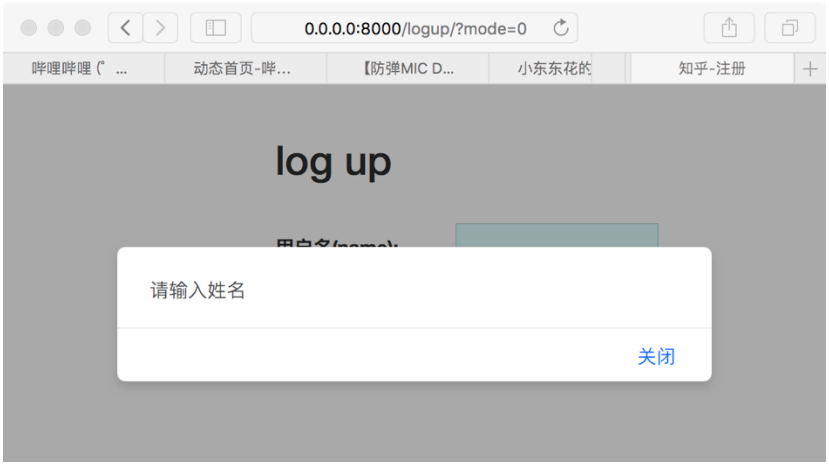
```
class Answer(models.Model):
    question = models.ForeignKey(Question, on_delete=models.CASCADE)#回答的问题
    user= models.ForeignKey(User, on_delete=models.CASCADE)#回答的用户
    content = models.CharField(max_length=150)#回答的内容
    img = models.ImageField(upload_to='upload', null=False, blank=True, default = os.path.join ( '/','upload','empty.png'),)#回答附加的图片
```

扩展用户模型

```
class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)#和系统用户模型关联
    focusedquestion = models.ManyToManyField(Question)#用户关注问题
    introduction = models.CharField(max_length=150)#用户介绍
```

1.注册

a)



使用 js 文件

判断是否输入用户名，判断密码是否符合要求，判断两次密码是否相同

views 方法

```
def logup(request):

    if request.method == 'GET':

        mode = request.GET['mode']
        return render(request, "logup.html", {'mode':mode})
    elif request.method == 'POST':
        username = request.POST['name']
        password = request.POST['pswd']
```

```
        num_results = User.objects.filter(username=username).count()
        if num_results != 0:

            return                                                  render(request,
"logup.html",{'mode':num_results,'existname':username})

        else:


User.objects.create_user(username=username,email=None,password=password)
            user= User.objects.get(username=username)
            profile = UserProfile()  # e**************************
            profile.user_id = user.id
            profile.save()
            return render(request, "login.html")
```

在第一跳转到注册时，mode 变量为 0

```
href="/logup/?mode=0"
```

提交注册表单后如果发现用户已经存在，mode 变量为 1 重新加载页面。

html 中通过提交的 mode 显示出用户名存在提示。

```
{% if mode == 1%}
<th><span    style="color:red"  id="existname"  >用户名{{existname}}已存在
</span></th>
{% endif %}
```

提交注册表单后发现用户名不冲突，创建系统自带用户对象，创建扩展用户对象，设置关联的系统用户。

2. 登录

js 文件

判断是否输入用户名，判断密码是否符合要求

views 方法

```
def loginView(request):

    if request.method == 'GET':

        return render(request, "login.html", {'mode':0})#首次登录
    elif request.method == 'POST':
        username = request.POST['name']
        password = request.POST['pswd']
        user = authenticate(username = username, password = password)

        if user is not None:#已经有这个用户了
            login(request,user)
           return HttpResponseRedirect("/home/")
        else:
            return render(request, "login.html", {'mode':1})#登录失败
```

在打开登录时，mode 变量为 0。

提交注册表单后如果发现登录信息错误，mode 变量为 1 重新加载页面。

html 中通过提交的 mode 显示出登录出错提示。

```
{% if mode == 1%}
<th><span  style="color:red" id="existname" >用户名不存在或密码错误</span></th>
{% endif %}
```

提交注册表单后发现用户名不冲突，通过重定向打开主页。

3. 退出

点击后显示登录界面

views 方法

```
@login_required
def logoutView(request):
    logout(request)
    return HttpResponseRedirect("/login/")
```

使用系统自带的 logout 函数，重定向到登录页面。

4. 首页 显示全部问题（根据回答数排列，前十个为热点问题）

views 方法

```
@login_required
def home(request):
    if request.method == 'GET':
        userprofile = UserProfile.objects.get(user=request.user)
        for question in Question.objects.all():
            if userprofile.focusedquestion.filter(id=question.id).count()
== 0:
                question.focus = 0
                question.save()
            else:
                question.focus = 1
                question.save()
        hotquestionlist = Question.objects.order_by('-
answerNum')[:10]#0108ying
        questionlist = Question.objects.order_by('-
answerNum')[11:]#0108ying
        #需要替换成真正首页
        return render(request, "home.html", {"hotquestionlist":
hotquestionlist,"questionlist": questionlist})#0108ying
    elif request.method == 'POST':
        title=request.POST['question_to_add']
        Question.objects.create(title=title,user=request.user)
        return HttpResponseRedirect("/home/")
```

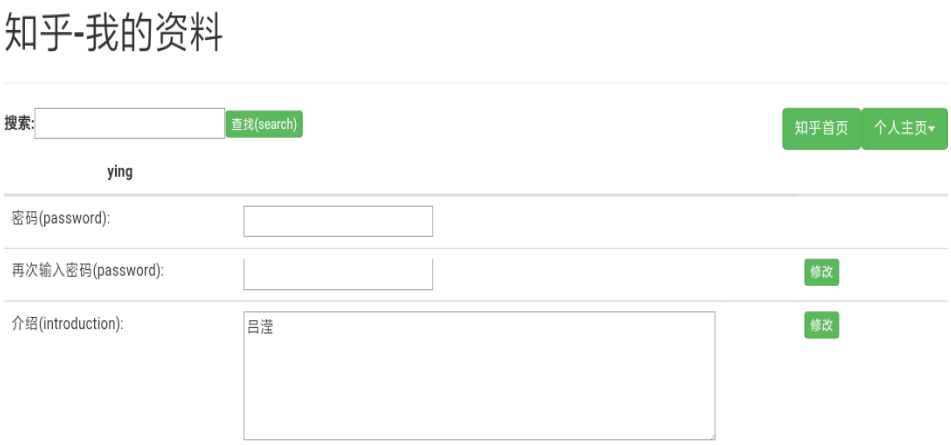
如果以 get 方式打开首页，根据用户的关注信息，遍历所有的问题，设置问题是否被当前用户关注，将问题根据回答数分为热点问题和普通问题并排序。

在 html 中，根据问题是否被关注，显示关注按钮

```
<a type="button"  class="btn btn-xs btn-success"
href="/question/?questionid={{ question.id }}">查看</a>
{% if question.focus == 0 %}
    <a type="button" id="{{question.id}}" class="btn btn-xs btn-info"
href="/focusquestion/?questionid={{ question.id }}">关注</a>
{% else %}
    <a type="button" id="{{question.id}}" class="btn btn-xs btn-warning" >已关
注</a>
{% endif %}
```

如果以 post 方式打开首页，是提交问题，将问题的标题，和当前登录用户保存到创建的问题对象。通过重定向，重新加载主页。

5.用户管理（查看个人主页 更改用户密码介绍）



js 文件

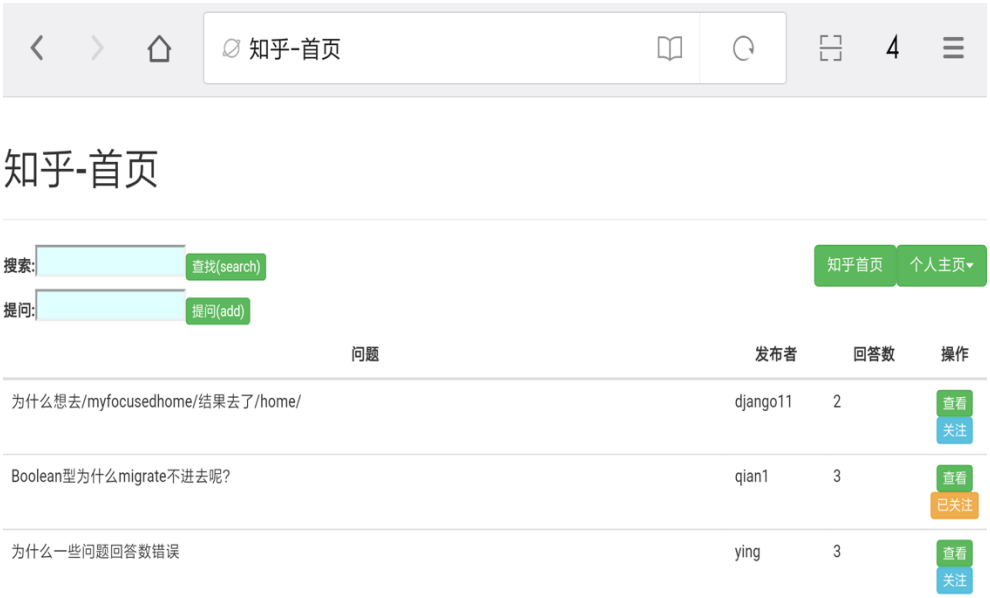
判断密码是否符合要求，判断两次密码是否相同。

views 方法

```
@login_required
def myprofile(request):
```

```
    if request.method == 'GET':
        userprofile = UserProfile.objects.get(user=request.user)
        return render(request, "myprofile.html", {"userprofile": userprofile})
@login_required
def editpassword(request):
    if request.method == 'POST':
        userprofile = UserProfile.objects.get(user=request.user)
        password = request.POST['pswd']
        user=request.user
        user.set_password(password)
        user.save()
        return HttpResponseRedirect("/myprofile/")

@login_required
def editprofile(request):
    if request.method == 'POST':
        newintroduction = request.POST['introduction']
        userprofile = UserProfile.objects.get(user=request.user)
        userprofile.introduction=newintroduction
        userprofile.save()
        return HttpResponseRedirect("/myprofile/")
```

6. 搜索问题



js

判断搜索内容不为空。

views 方法

```
@login_required()
def search(request):
    if request.method == 'POST':
        content=request.POST["question_to_search"]
        questionlist=Question.objects.filter(title__contains=content)
        return render(request,"home.html",{"questionlist":questionlist})
```

通过 filter 的属性 title__contains 得到所有含有搜索关键字的问题。

7. 我的问题 （查看列表）

views 查看问题列表

```
@login_required
def myhome(request):
    if request.method == 'GET':
```

```
        questionlist =
Question.objects.filter(user=request.user).order_by('-id')
        return render(request, "myhome.html", {"questionlist":
questionlist})
    elif request.method == 'POST':
        title=request.POST['question_to_add']
        Question.objects.create(title=title,user=request.user)
        return HttpResponseRedirect("/myhome/")
```

和首页的方法类似

html

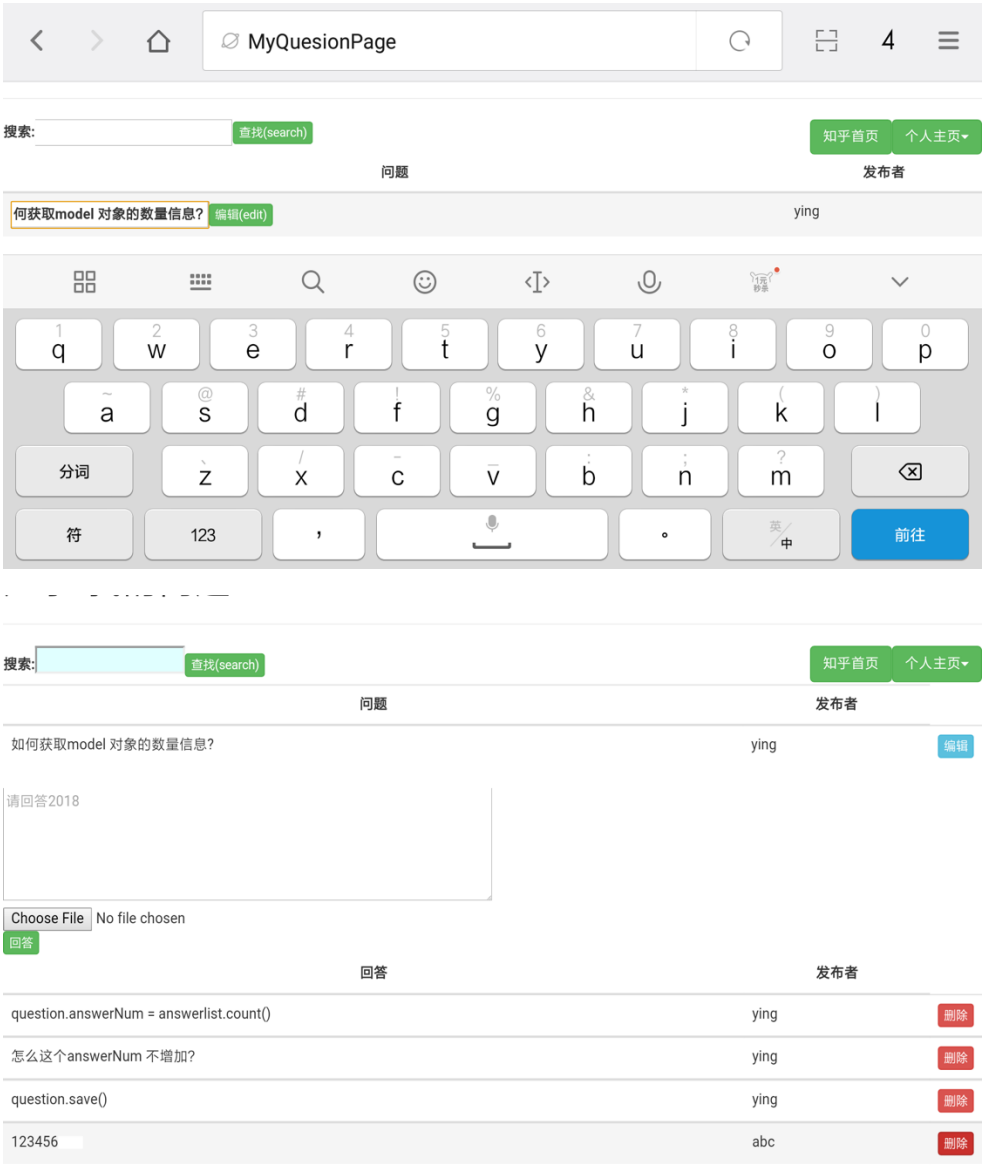
```
<a type="button" class="btn btn-xs btn-success"
href="/myquestion/?questionid={{ question.id }}">查看</a>
<a type="button" class="btn btn-xs btn-info"
href="/questionadminedit/?questionid={{ question.id }}">编辑</a>
<a type="button" class="btn btn-xs btn-danger"
href="/questiondel/?questionid={{ question.id }}">删除</a>
```

每个问题都会有对应的查看、编辑、删除链接，都是作为问题作者的权限打开。

views 删除自己的问题

```
@login_required#
def questiondel(request):
    questionid = request.GET['questionid']#获取删除条目的编号
    if Question.objects.filter(id = questionid).count() == 0:#防止多次请求
        return HttpResponseRedirect("/myhome/")
    Question.objects.get(id=questionid).delete()
    return HttpResponseRedirect("/myhome/")
```

8.我的问题 （编辑问题）



views 查看我的问题的回答列表

```
@login_required
def myquestion(request):
    if request.method == "GET":
        questionid=request.GET["questionid"]
        question=Question.objects.get(id=questionid)
        question.mode = False
        question.save()
        answerlist=Answer.objects.filter(question=question)
        return
render(request,"myquestion.html",{"question":question,"answerlist":answerli
st})
    elif request.method == "POST":#
        content=request.POST['content']
        questionid=request.POST['questionid']
```

```
        question=Question.objects.get(id=questionid)



newanswer=Answer.objects.create(user=request.user,question=question,content
=content)
        pic = request.FILES.get("pic")
        if not not pic:
            newanswer.imgname = request.FILES['pic'].name
            newanswer.img =request.FILES['pic']
            newanswer.save()
        answerlist=Answer.objects.filter(question=question)
        question.answerNum = answerlist.count() # 0108ying
        question.save()# 0108ying
        return
render(request,'myquestion.html',{"question":question,"answerlist":answerli
st})
```

get 方法筛选出属于这个问题的回答，post 方法可以对自己问题回答。

views 编辑我的问题

```
@login_required
def questionadminedit(request):
    if request.method == 'GET':
        questionid = request.GET['questionid']
        question = Question.objects.get(id = questionid)
        answerlist = Answer.objects.filter(question = question )
        return
render(request,"editquestion.html",{"question":question,"answerlist":answerlist
})
    elif request.method == 'POST':
        newtitle = request.POST['new_title']
        questionid = request.POST['questionid']
        question = Question.objects.get(id = questionid)


        question.title  = newtitle
        question.save()
        answerlist = Answer.objects.filter(question=question)
        return render(request, 'myquestion.html', {"question": question,
"answerlist": answerlist})
```

在 editquestion.html 中会显示出原始问题，方便编辑。

```
<th><input type="text" name="new_title"  id="question"
value="{{question.title}}" size="20"></th>
```

post 后返回我的这个问题。

9. 我的问题 （删除回答）

回答	发布者	
question.answerNum = answerlist.count()	ying	删除
怎么这个answerNum 不增加?	ying	删除
question.save()	ying	删除
123456	abc	删除

回答	发布者	
question.answerNum = answerlist.count()	ying	删除
怎么这个answerNum 不增加?	ying	删除
question.save()	ying	删除

```
@login_required#
def answeradmindel(request):
    answerid = request.GET['answerid']#获取删除条目的编号
    question=Answer.objects.get(id=answerid).question
    if Answer.objects.filter(id = answerid).count() == 0:#防止多次请求
        answerlist = Answer.objects.filter(question=question)
        return render(request, 'myquestion.html', {"question": question,
"answerlist": answerlist})
    Answer.objects.get(id=answerid).delete()
    answerlist = Answer.objects.filter(question=question)
    question.answerNum = answerlist.count()  # 0108ying
    question.save()  # 0108ying
    thisquestionid= question.id
    return render(request, 'myquestion.html', {"question": question,
"answerlist": answerlist})
```

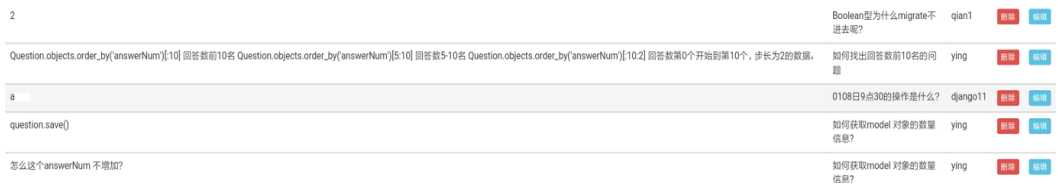
在删除回答后，由于回答数发生改变，所以更新了 question.answerNum

在测试过程中，一次删除问题后，点击了浏览器的后退键，之后再次点击删除，网页报错。我根据错误报告了解到，之前的回答已经从数据库中删除，再次删除时找不到相应回答。为了增强网页健壮性，我增加了一步判断。

10. 我的回答

问题	发布者
看看手机能上吗	qian1
性能测试截图	qian1
啦啦啦	django11

我的资料
我的问题
我的回答
我的关注
新消息
退出



删除后

views 回答列表

```
@login_required
def myanswer(request):
    if request.method == 'GET':
        answerlist = Answer.objects.filter(user=request.user).order_by('-id')
        return render(request,"myanswer.html",{"answerlist":answerlist})
```

根据用户名筛选出回答。

html 回答列表，每个回答显示出内容图片，问题，提问的人，删除，编辑

```
<td>{{ answer.content}}<img  src="/static/media/{{answer.img.name}}"
alt="{{answer.img.url}}"></td>
<td>{{ answer.question.title}}</td>
<td>{{ answer.question.user}}</td>
<td><a type="button" class="btn btn-xs btn-danger"
href="/answerdel/?answerid={{ answer.id }}">删除</a></td>
<td><a type="button" class="btn btn-xs btn-info"
href="/answeredit/?answerid={{ answer.id }}">编辑</a></td>
```

views 删除我的回答

```
@login_required
def answerdel(request):
    answerid = request.GET['answerid']
    if Answer.objects.filter(id = answerid).count() == 0:#防止多次请求
        return HttpResponseRedirect("/myanswer/")
    answer = Answer.objects.get(id = answerid)
    question = answer.question
    answer.delete()
    answerToThis = Answer.objects.filter(question=question)
    question.answerNum = answerToThis.count()  # 0108ying
    question.save()  # 0108ying
    return HttpResponseRedirect("/myanswer/")
```

删除回答后，问题的回答数变化，需要更新。

在测试过程中，一次删除问题后，点击了浏览器的后退键，之后再次点击删除，网页报错。我根据错误报告了解到，之前的回答已经从数据库中删除，再次删除时找不到相应回答。为了增强网页健壮性，我增加了一步判断。

views 编辑我的回答

```
@login_required
def answeredit(request):
    if request.method == 'GET':
        answerid = request.GET['answerid']
        answer = Answer.objects.get(id = answerid)
        question = answer.question
        return
render(request,"editanswer.html",{"question":question,"answer":answer})
    elif request.method == 'POST':
        content = request.POST['content']
        answerid = request.POST['answerid']
        answer = Answer.objects.get(id = answerid)
        answer.content = content
        pic = request.FILES.get("pic")
        if not not pic:
            answer.imgname = request.FILES['pic'].name
            answer.img =request.FILES['pic']
        answer.save()
        question = answer.question
        if question.user != request.user: #自己回答自己的问题不用提醒
            question.mode = True
        question.save()
        return HttpResponseRedirect("/myanswer/")
```

编辑回答时，html 文件会将之前的内容显示出来，仅对细节编辑更加方便。同时也可以选择取消之前的图片，或者更换新的图图片。

2. 新消息

a) 效果

另 一 用 户 回 答 我 的 问 题

知乎-我的问题

搜索: 查找(search) 知乎首页 个人主页

问题	发布者	
如何获取model 对象的数量信息?	ying	编辑

新消息尝试。

选取文件 未选择文件

回答

回答	发布者	
question.answerNum = answerlist.count()	ying	删除
怎么这个answerNum 不增加?	ying	删除
question.save()	ying	删除

知乎-问题

搜索: 查找(search) 知乎首页 个人主页

问题	发布者
如何获取model 对象的数量信息?	ying

请回答2018

选取文件 未选择文件

回答

回答	发布者
question.answerNum = answerlist.count()	ying
怎么这个answerNum 不增加?	ying
question.save()	ying
新消息尝试。	django4

我查看新消息

我浏览有新消息的问题

再次查看新消息列表

对于不是自己的问题，在编辑回答和增加新回答时，会更改问题的状态。

```
if question.user != request.user: #自己回答自己的问题不用提醒
```

```
        question.mode = True
```

查看一个问题即用 get 方式打开 question 时，

```
if question.user == request.user:
    question.mode = False
    question.save()
```

如果问题是自己的，将问题状态设置为已读。

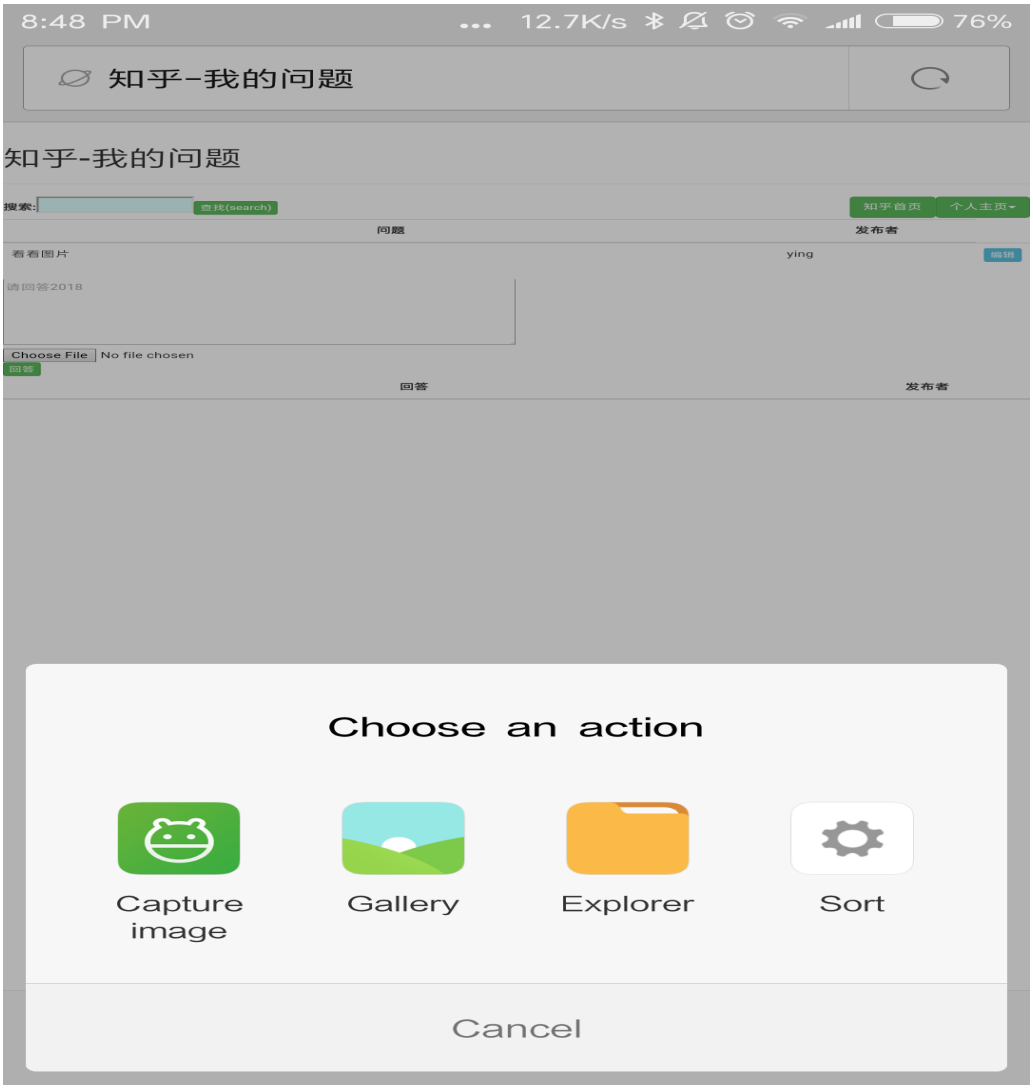
views 新消息

```
@login_required()
def mynewanswer(request):
    if request.method == "GET":
        questionlist = Question.objects.filter(user = request.user, mode = True).order_by('-id')
        return render(request,"myhome.html",{"questionlist":questionlist})
```

显示所有设置为有提示的问题。

11. 添加图片

b) 效果




8:49 PM
4.53K/s
76%
知乎-我的问题
知乎-我的问题
搜索:
查找(search)
知乎首页
个人主页
问题
发布者
看看图片
ying
编辑
请回答2018
Choose File
tpo32_writing.jpg
回答
回答
发布者
4

知乎-我的问题

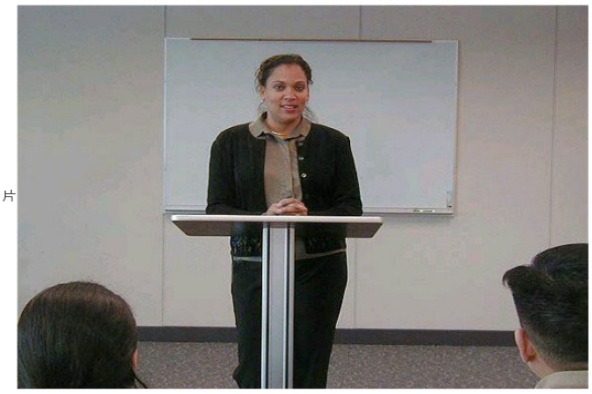
搜索: 查找(search) 知乎首页 个人主页

问题	发布者	
看看图片	ying	编辑

请回答2018

Choose File No file chosen

回答

回答	发布者	
一张图片 	ying	删除

views 提交图片

```
pic = request.FILES.get("pic")
if not not pic:
    newanswer.imgname = request.FILES['pic'].name
    newanswer.img =request.FILES['pic']
    newanswer.save()
```

settings 设置路径

```
MEDIA_ROOT = 'default/static/media/'

MEDIA_URL = 'default/static/media/'
```

html 的表单增加属性

```
<form action="/question/" method="post" enctype="multipart/form-data">
```

（四） 测试

使用 ab 在命令行中测试 myanswer 页面的性能，结果如下：

```
Ying — -bash — 80×56
[YingLvs-MacBook-Pro:~ Ying$ ab -n 1000 -c 10 http://0.0.0.0:8000/myanswer/
This is ApacheBench, Version 2.3 <$Revision: 1807734 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 0.0.0.0 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        WSGIServer/0.2
Server Hostname:        0.0.0.0
Server Port:            8000

Document Path:          /myanswer/
Document Length:        0 bytes

Concurrency Level:      10
Time taken for tests:   1.912 seconds
Complete requests:      1000
Failed requests:        0
Non-2xx responses:      1000
Total transferred:      234000 bytes
HTML transferred:       0 bytes
Requests per second:    522.95 [#/sec] (mean)
Time per request:       19.122 [ms] (mean)
Time per request:       1.912 [ms] (mean, across all concurrent requests)
Transfer rate:          119.50 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       2
Processing:     3   19   2.6     19      32
Waiting:        1   17   2.3     17      31
Total:          3   19   2.6     19      32

Percentage of the requests served within a certain time (ms)
  50%     19
  66%     19
  75%     20
  80%     21
  90%     22
  95%     23
  98%     26
  99%     27
 100%     32 (longest request)
YingLvs-MacBook-Pro:~ Ying$
```

YingLvs-MacBook-Pro:~ Ying$ ab -n 1000 -c 10 http://0.0.0.0:8000/question/?questionid=17

This is ApacheBench, Version 2.3 <$Revision: 1807734 $>

Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/

```
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking 0.0.0.0 (be patient)
Completed 100 requests
Completed 200 requests
Completed 300 requests
Completed 400 requests
Completed 500 requests
Completed 600 requests
Completed 700 requests
Completed 800 requests
Completed 900 requests
Completed 1000 requests
Finished 1000 requests


Server Software:        WSGIServer/0.2
Server Hostname:        0.0.0.0
Server Port:            8000

Document Path:          /question/?questionid=17
Document Length:        0 bytes

Concurrency Level:      10
/* 整个测试持续的时间 * /
Time taken for tests:   2.003 seconds
/* 完成的请求数量 */
Complete requests:      1000
/* 失败的请求数量 */
Failed requests:        0
Non-2xx responses:      1000
/* 整个场景中的网络传输量 */
Total transferred:      252000 bytes
/* 整个场景中的 HTML 内容传输量 */
HTML transferred:       0 bytes
/* 大家最关心的指标之一，相当于 LR 中的 每秒事务数 ，后面括号中的 mean 表示这是一个平均值 */
Requests per second:    499.30 [#/sec] (mean)
/* 大家最关心的指标之二，相当于 LR 中的 平均事务响应时间 ，后面括号中的 mean 表示这是一个平均值 */
Time per request:       20.028 [ms] (mean)
Time per request:       2.003 [ms] (mean, across all concurrent requests)
/* 平均每秒网络上的流量，可以帮助排除是否存在网络流量过大导致响应时间延长的问题 */
```

```
Transfer rate:          122.88 [Kbytes/sec] received
/* 网络上消耗的时间的分解*/
Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:     5   20   2.5     19      28
Waiting:        2   18   2.2     18      24
Total:          6   20   2.5     20      29

Percentage of the requests served within a certain time (ms)
  50%     20
  66%     21
  75%     21
  80%     22
  90%     23
  95%     24
  98%     25
  99%     26
 100%     29 (longest request)
YingLvs-MacBook-Pro:~ Ying$
```

（五） 问题分析

1.为什么想去/MYFOCUSEDHOME/结果去了/HOME/

因为 url 的匹配顺序出错，应该将/myfocusedhome/放在/home/前

2.如何将图片上传到服务器的文件夹

特别注意的是，只有当 request 方法是 POST，且发送 request 的<form>有属性 enctype="multipart/form-data"时，request.FILES 中包含文件数据，否则 request.FILES 为空。

--转自 django 中处理文件上传文件

https://www.cnblogs.com/zhaopengcheng/p/5633154.html

安装了 pillow 才能使用 ImageField

转自 讲解 http://blog.csdn.net/meylovezn/article/details/47124923

3.Boolean 型为什么 migrate 不进去？

要赋初值 mode = model.BooleanField(default=false)

4 如何使用 pycharm 自带的 django 运行按钮将网页部署在局域网？

setting.py 中

```
MIDDLEWARE = [ 'django.middleware.security.SecurityMiddleware',

    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
```

'django.middleware.clickjacking.XFrameOptionsMiddleware',]

替换掉 MIDDLEWARE_CLASS[] 的内容

5. 为什么一些问题回答数错误？

answerdel 筛选 有问题，筛除的是当前用户的回答数而不是问题的回答数。

314 行 focusedquestiondel 写成了 questiondel，覆盖了之前的 questiondel 图