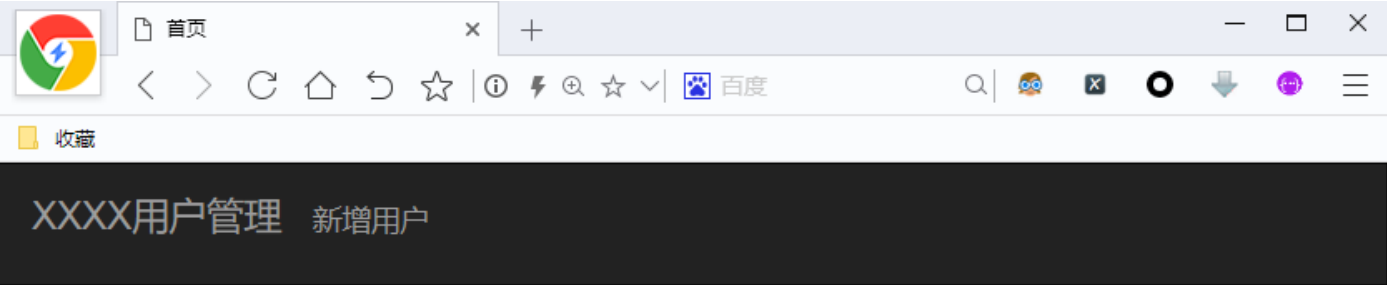


1 项目说明

制作一个用户信息管理模型 用户信息包括： - 姓名 - 密码 - 年龄 - 邮箱 - 创建时间

产品功能： 1. 全部用户信息的展示（查） 2. 添加新用户（增） 3. 修改用户信息（改） 4. 删除用户（删）

项目展示： 首页显示所有的用户数据：



id	用户名	密码	年龄	邮箱	创建时间	操作
1	乔峰	5659873	40	qiaofeng@csdn.net	2019-12-18 03:51:35	修改 删除
2	郭靖	2301460	30	guojing@csdn.net	2019-12-18 03:51:35	修改 删除
4	张无忌	1087875	20	zhangwuji@csdn.net	2019-12-18 03:51:35	修改 删除
5	王重阳	1650187	60	wangchongyang@csdn.net	2019-12-18 03:51:35	修改 删除
6	逍遥子	4486387	100	xiaoyaozi@csdn.net	2019-12-18 03:51:35	修改 删除
7	张三丰	4340150	95	zhangsanfeng@csdn.net	2019-12-18 03:51:35	修改 删除
8	独孤求败	7883464	30	duguqiubai@csdn.net	2019-12-18 03:51:35	修改 删除
9	东方不败	8550976	25	dongfangbubai@csdn.net	2019-12-18 03:51:35	修改 删除
10	扫地僧	6110380	66	saodiseng@csdn.net	2019-12-18 03:51:35	修改 删除
11	小龙女	123456	18	xiaolongnv@csdn.net	2019-12-18 05:26:41	修改 删除

新增用户：

新标签页

新增用户

+

—

□

×



 百度









收藏

XXXX用户管理 新增用户

用户名

请输入用户名

密码

请输入密码

年龄

请输入年龄

邮箱

请输入邮箱

添加

修改用户：



用户名

密码

年龄

邮箱

修改

删除用户:

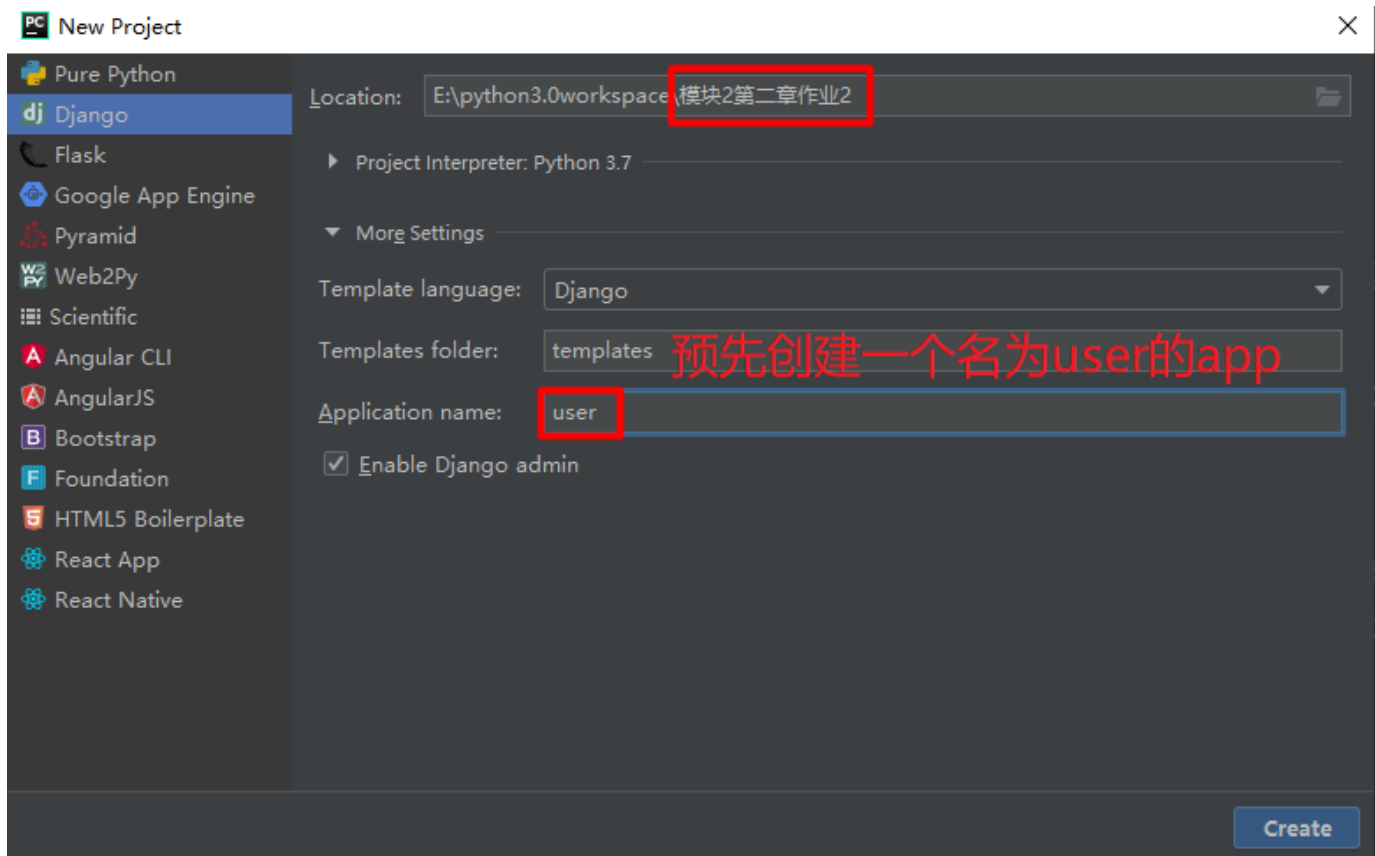


删除用户成功

关于前端： - 前端页面布局与样式不是课程重点，在资料中有静态文件代码，是已经为大家准备好的静态HTML

2 搭建项目

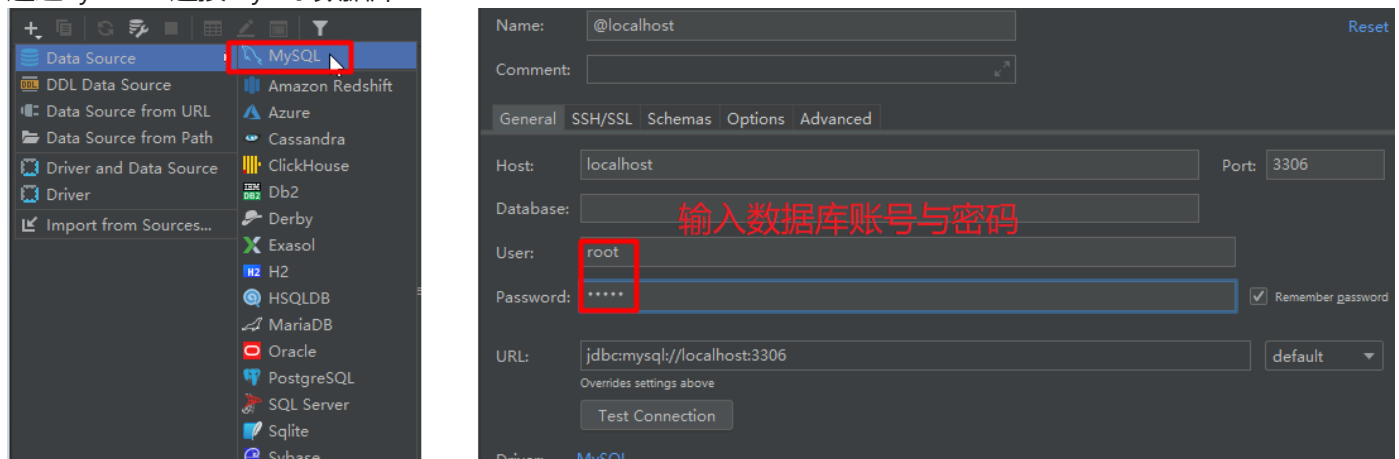
2.1 创建Django项目



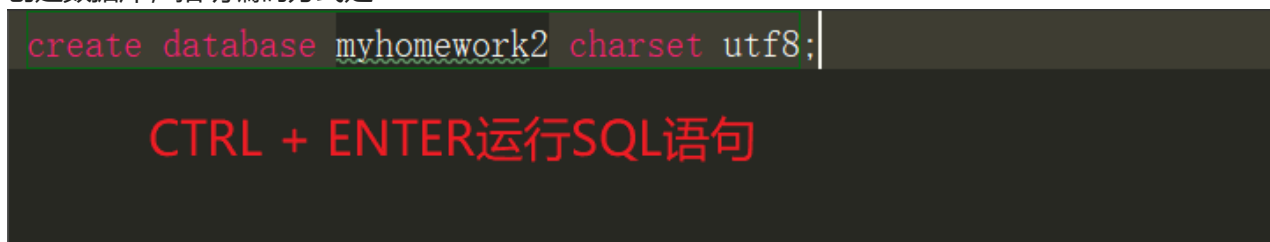
2.2 数据库设计

2.2.1 数据库配置

通过Pycharm连接MySQL数据库



创建数据库，指明编码方式是utf8



在settings中配置刚刚创建好的数据库

▼ 模块2第二章作业2 E:\python3.0\

templates

▶ user

▼ 模块2第二章作业2

__init__.py

settings.py

urls.py

wsgi.py

manage.py

External Libraries

Scratches and Consoles

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

Database

<https://docs.djangoproject.com/en/2.2/ref/settings/#>

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'HOST': 'localhost',

'PORT': '3306',

'NAME': 'myhomework2',

'USER': 'root',

'PASSWORD': '12345',

}

}

2.2.2 字段分析

根据项目要求，用户的信息包括：用户名、密码、年龄、邮箱、创建时间

字段含义	字段关键字	类型	约束
用户名	username	varchar(16)	unique,not null
密码	password	varchar(32)	not null
年龄	age	tinyint	not null
邮箱	email	varchar(80)	
创建时间	createDatetime	datetime	default=now()

2.2.3 定义模型类

▼ 模块2第二章作业2 E:\python3.0\

templates

▼ user

migrations

__init__.py

admin.py

apps.py

models.py

tests.py

views.py

模块2第二章作业2

manage.py

External Libraries

Scratches and Consoles

1

2

3

4

5

6

7

8

9

10

11

from django.db import models

import datetime

Create your models here.

class User(models.Model):

username = models.CharField(max_length=16,unique=True)

password = models.CharField(max_length=32)

age = models.SmallIntegerField()

email = models.CharField(max_length=80)

createDatetime = models.DateTimeField(auto_now_add=datetime.datetime.now())

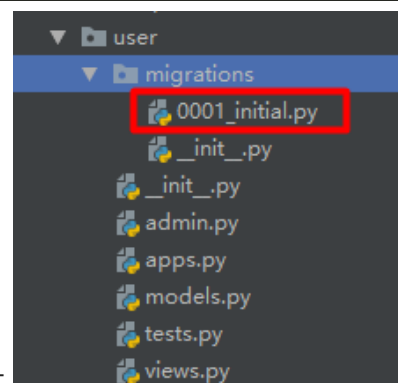
```
1 from django.db import models
2 import datetime
3
4 # Create your models here.
5
6 class User(models.Model):
7     username = models.CharField(max_length=16, unique=True)
8     password = models.CharField(max_length=32)
9     age = models.SmallIntegerField()
10    email = models.CharField(max_length=80)
11    createDatetime = models.DateTimeField(auto_now_add=datetime.datetime.now())
```

2.3.4 数据迁移

2.3.4.1 生成迁移文件

```
1 python manage.py makemigrations
```

```
E:\python3.0workspace\模块2第二章作业2>python manage.py makemigrations
```



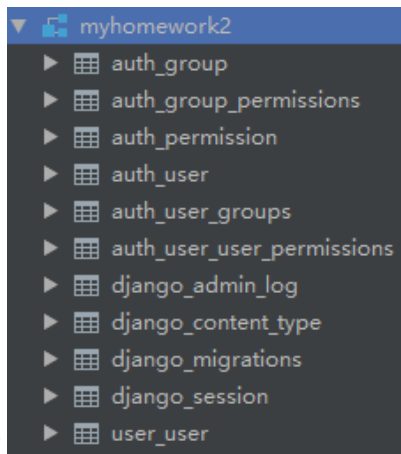
运行后，在User app的migrations文件夹中出现数据迁移文件

2.3.4.2 执行迁移文件

```
1 python manage.py migrate
```

E:\python3.0workspace\模块2第二章作业2>python manage.py migrate

得到更新



后的数据库

2.3 构建路由与视图

2.3.1 路由分析

需要的页面有： - 首页：展示所有用户数据 - 添加新用户：表单实现数据的提交 - 修改用户：根据Id对应被修改的用户，表单数据实现提交 - 删除用户：发送被删除用户的Id到服务器，完成删除 - 操作结果：显示添加、修改与删除操作是否成功

路由传参分析： - 首页：展示所有的用户信息，不需要接受参数 - 添加新用户：新增的用户数据由表单提交，不需要路由传参 - 修改用户：需要传递被修改用户的Id，Id由路由接收，修改的用户数据又表单提交 - 删除用户：由路由接受被删除的用户Id，后台完成删除操作

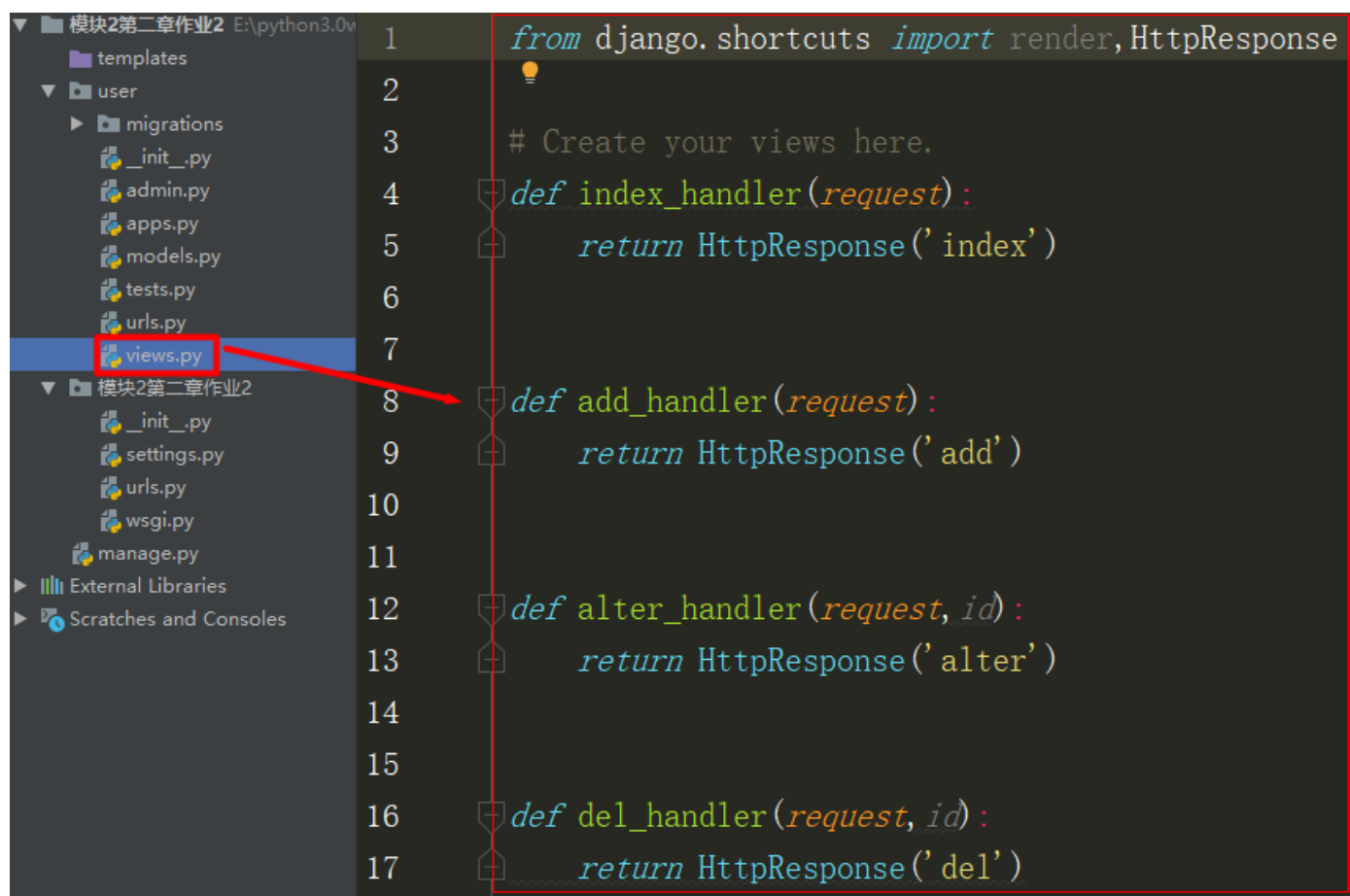
2.3.2 user中的子路由定义

```
1 from django.urls import path, re_path
2 from . import views
3
4 urlpatterns = [
5     path('', views.index_handler, name='index'), # 数据展示
6     path('add', views.add_handler, name='add'), # 数据添加
7     re_path('alter/(\d+)', views.alter_handler, name='alter'), # 根据Id进行修改
8     re_path('del/(\d+)', views.del_handler, name='del'), # 根据Id进行删除
9 ]
```



2.3.3 创建视图函数

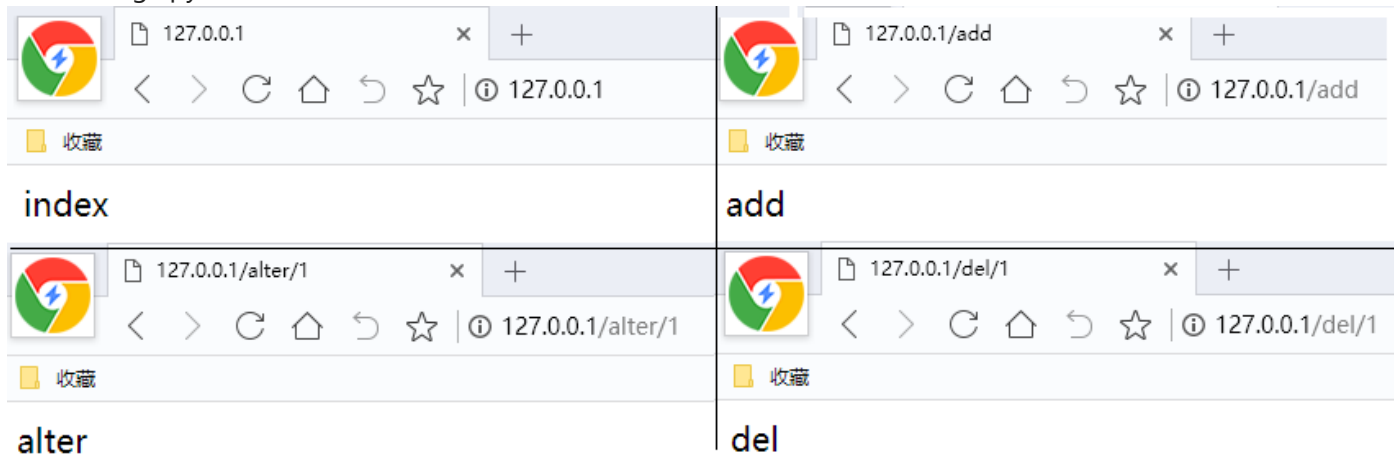
```
1 from django.shortcuts import render,HttpResponse
2
3 # Create your views here.
4 def index_handler(request):
5     return HttpResponse('index')
6
7
8 def add_handler(request):
9     return HttpResponse('add')
10
11
12 def alter_handler(request,id):
13     return HttpResponse('alter')
14
15
16 def del_handler(request,id):
17     return HttpResponse('del')
```



2.3.4 主路由中包含子路由

2.4 测试项目

运行manage.py，检查所有路由与视图是否正常运行



3 把静态HTML整理成模板

3.1 迁移HTML

找到作业资料中为大家准备的静态HTML文件

add.html	2019/12/17 11:49	TSBrowser HTM...	3 KB
alter.html	2019/12/17 11:50	TSBrowser HTM...	3 KB
index.html	2019/12/17 11:48	TSBrowser HTM...	4 KB
result.html	2019/12/17 11:50	TSBrowser HTM...	1 KB

注：这个几个静态文

件可以直接拖拽到浏览器中运行，打开后的结果如下： 首页：

XXXX用户管理

新增用户

id	用户名	密码	年龄	邮箱	创建时间	操作
1	叶良辰	123456	20	123@qq.com	2019-12-16 00:00:01	修改 删除
1	叶良辰	123456	20	123@qq.com	2019-12-16 00:00:01	修改 删除
1	叶良辰	123456	20	123@qq.com	2019-12-16 00:00:01	修改 删除
1	叶良辰	123456	20	123@qq.com	2019-12-16 00:00:01	修改 删除
1	叶良辰	123456	20	123@qq.com	2019-12-16 00:00:01	修改 删除
1	叶良辰	123456	20	123@qq.com	2019-12-16 00:00:01	修改 删除

新增用户页：

XXXX用户管理

新增用户

用户名

请输入用户名

密码

请输入密码

年龄

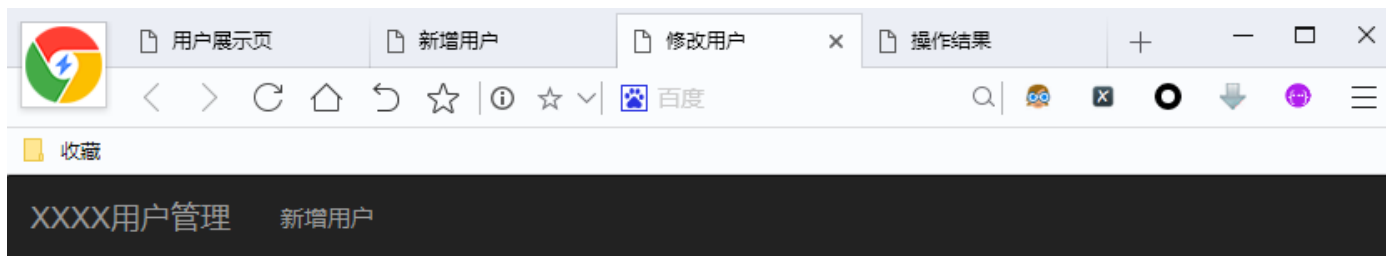
请输入年龄

邮箱

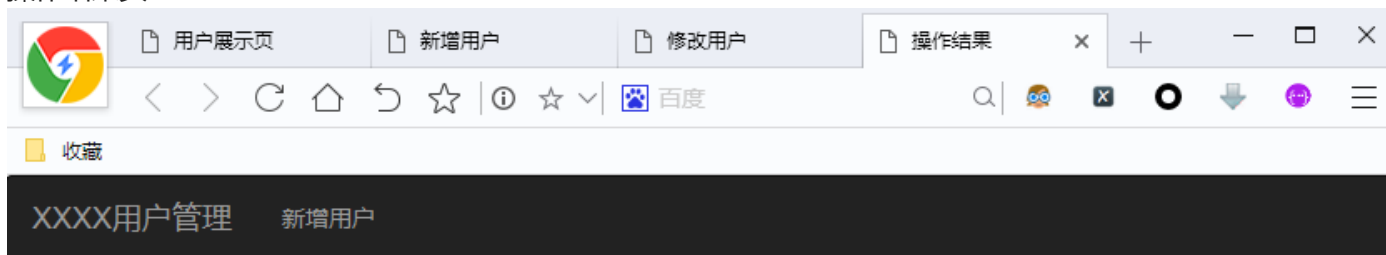
请输入邮箱

添加

修改用户页：



操作结果页：



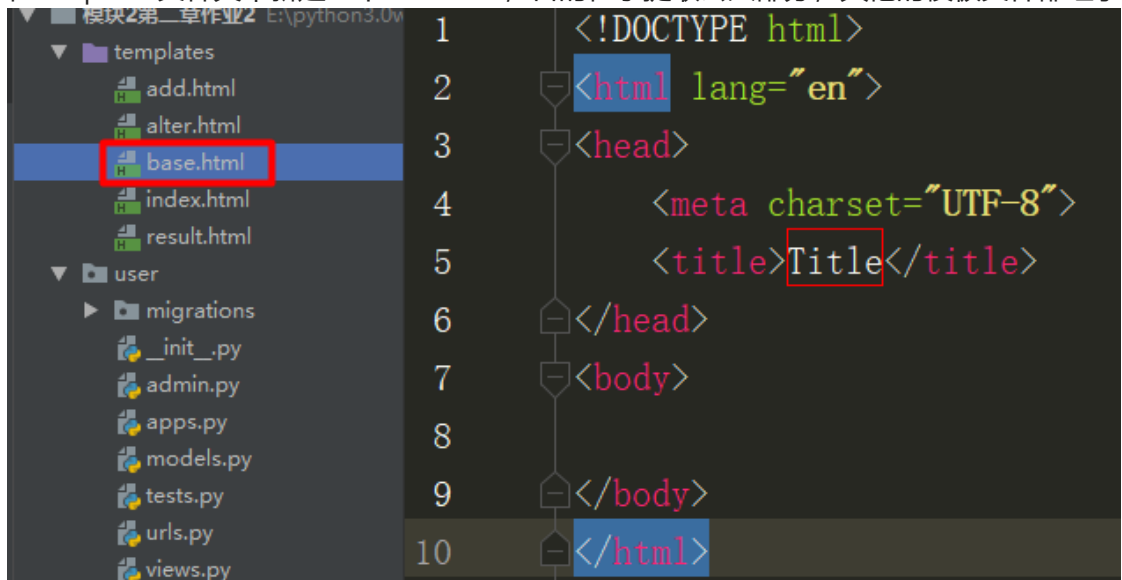
操作成功

把这几个HTML粘贴到templates模板文件夹中

3.2 模板继承与代码块提取

3.2.1 建立父模板

在templates文件夹中新建一个base.html，目的在于提取公共部分，其他的模板文件都继承base.html



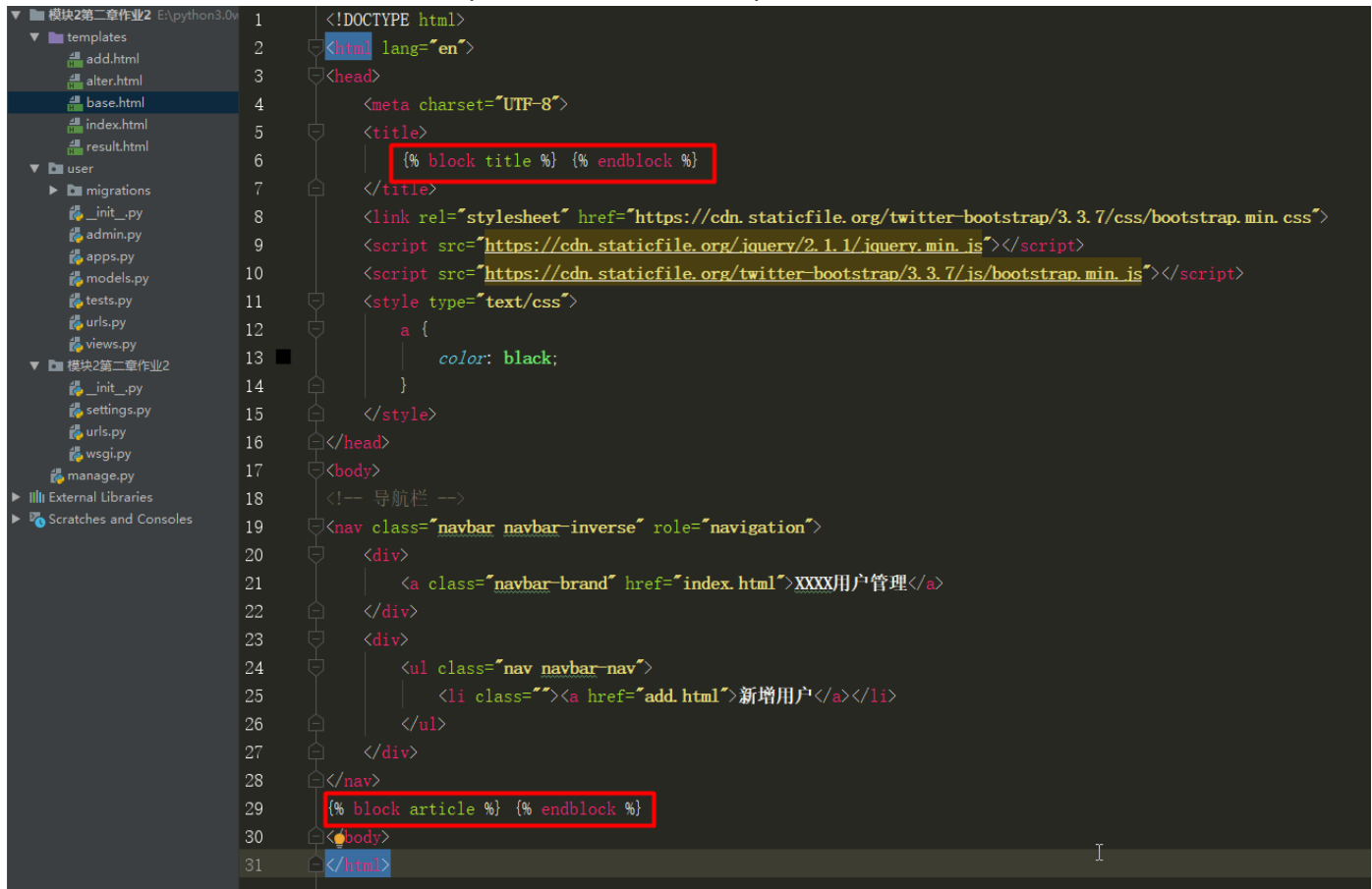
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>Title</title>
6 </head>
7 <body>
8
9 </body>
10 </html>
```

把index.html中的内

容复制粘贴到base.html中

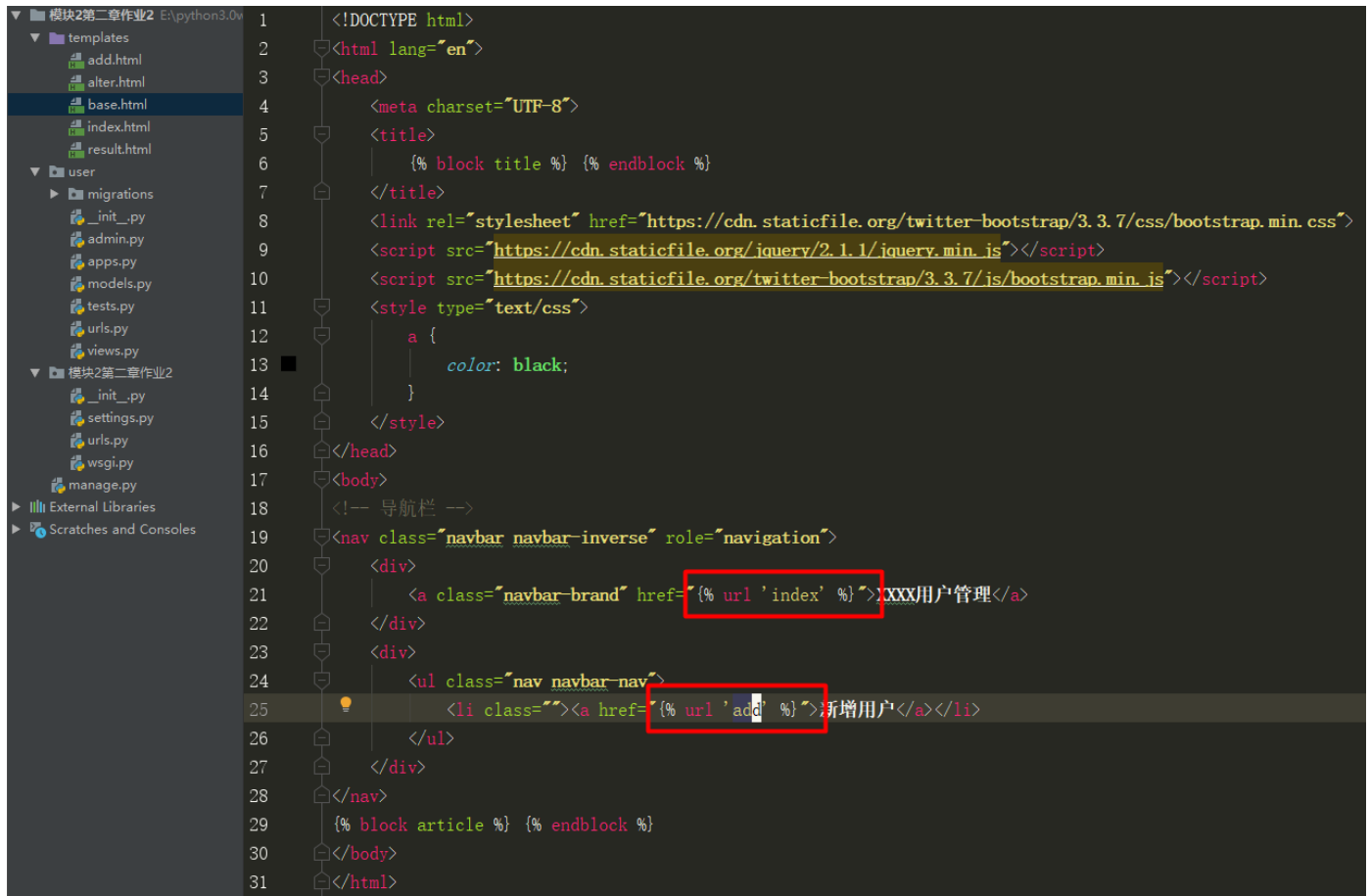
3.2.2 代码块抽离

抽离出两个代码块： 1. title 2. article （导航栏下面的主题部分）



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     <meta charset="UTF-8">
5     <title>
6         {% block title %} {% endblock %}
7     </title>
8     <link rel="stylesheet" href="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/css/bootstrap.min.css">
9     <script src="https://cdn.staticfile.org/jquery/2.1.1/jquery.min.js"></script>
10    <script src="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>
11    <style type="text/css">
12        a {
13            color: black;
14        }
15    </style>
16 </head>
17 <body>
18     <!-- 导航栏 -->
19     <nav class="navbar navbar-inverse" role="navigation">
20         <div>
21             <a class="navbar-brand" href="index.html">XXXX用户管理</a>
22         </div>
23         <div>
24             <ul class="nav navbar-nav">
25                 <li class=""><a href="add.html">新增用户</a></li>
26             </ul>
27         </div>
28     </nav>
29     {% block article %} {% endblock %}
30 </body>
31 </html>
```

修改导航栏中的两个链接，通过反向解析路由，得到链接地址



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>
6     {% block title %} {% endblock %}
7   </title>
8   <link rel="stylesheet" href="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/css/bootstrap.min.css">
9   <script src="https://cdn.staticfile.org/jquery/2.1.1/jquery.min.js"></script>
10  <script src="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script>
11  <style type="text/css">
12    a {
13      color: black;
14    }
15  </style>
16 </head>
17 <body>
18  <!-- 导航栏 -->
19  <nav class="navbar navbar-inverse" role="navigation">
20    <div>
21      <a class="navbar-brand" href="{% url 'index' %}">XXXX用户管理</a>
22    </div>
23    <div>
24      <ul class="nav navbar-nav">
25        <li class=""><a href="{% url 'add' %}">新增用户</a></li>
26      </ul>
27    </div>
28  </nav>
29  {% block article %} {% endblock %}
30 </body>
31 </html>
```

完整代码: `python <!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <title> {% block title %} {% endblock %} </title> <link rel="stylesheet" href="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/css/bootstrap.min.css"> <script src="https://cdn.staticfile.org/jquery/2.1.1/jquery.min.js"></script> <script src="https://cdn.staticfile.org/twitter-bootstrap/3.3.7/js/bootstrap.min.js"></script> <style type="text/css"> a { color: black; } </style> </head> <body> <!-- 导航栏 --> <nav class="navbar navbar-inverse" role="navigation"> <div> XXXX用户管理 </div> <div> <ul class="nav navbar-nav"> <li class="">新增用户 </div> </nav> {% block article %} {% endblock %} </body> </html>`

3.3 index继承base

```

1 {% extends 'base.html' %}
2
3 {% block title %} 首页 {% endblock %}
4
5 {% block article %}
6     <!-- 数据展示 -->
7     <div class="container">
8         <table class="table table-hover">
9             <!-- 表头 -->
10            <thead...>
21            <!-- 表内容 -->
22            <tbody>
23                <tr...>
35            </tbody>
36        </table>
37    </div>
38 {% endblock %}

```

完整代码：

```
`python {% extends 'base.html' %}
```

```
{% block title %} 首页 {% endblock %}
```

```
{% block article %}
```

id	用户名	密码	年龄	邮箱	创建时间	操作
1	叶良辰	123456	20	123@qq.com	2019-12-16 00:00:01	修改 删除

```
{% endblock %} `
```

3.4 add继承base

```

1 {% extends 'base.html' %}
2
3 {% block title %} 新增用户 {% endblock %}
4
5 {% block article %}
6     <!-- 表单内容 -->
7     <div class="col-sm-offset-3 col-sm-6">
40
41 {% endblock %}

```

完整代

码：`python {% extends 'base.html' %}

```
{% block title %} 新增用户 {% endblock %}
```

```
{% block article %}
```

用户名

密码

年龄

邮箱

{% endblock %}`

3.5 alter继承base

▼ 模块2第二章作业2 E:\python3.0v

▼ templates

add.html

alter.html

base.html

index.html

result.html

▼ user

► migrations

__init__.py

admin.py

```
1 {% extends 'base.html' %}
2
3 {% block title %} 修改用户 {% endblock %}
4
5 {% block article %}
6     <!-- 表单内容 -->
7     <div class="col-sm-offset-3 col-sm-6">...</div>
40 {% endblock %}
41
```

完整代

码：`python {% extends 'base.html' %}

{% block title %} 修改用户 {% endblock %}

{% block article %}

用户名

密码

年龄

邮箱

{% endblock %}`

3.6 result继承base

```
▼ 模块2第二章作业2 E:\python3.0v
  ▼ templates
    add.html
    alter.html
    base.html
    index.html
    result.html
  ▼ user
    migrations
    _init_.py

1  {% extends 'base.html' %}
2  {% block title %} 操作结果 {% endblock %}
3  {% block article %}
4      <div class="col-sm-offset-4 col-sm-4">
5          <h2>操作成功</h2>
6      </div>
7  {% endblock %}
8
```

完整代码：p

```
python {% extends 'base.html' %} {% block title %} 操作结果 {% endblock %} {% block article %} <div class="col-sm-offset-4 col-sm-4"> <h2>操作成功</h2> </div> {% endblock %}
```

4 完善功能

4.1 首页

4.1.1 创建初始数据

因为首页是数据展示页，我们需要一些用户的数据，可以预先自己在user表中随意添加一些数据 例如，这里是我创建的10个初始用户数据：

<Filter criteria>						
	id	username	password	age	email	createDatetime
1	1	乔峰	5659873	35	qiaofeng@csdn.net	2019-12-18 03:51:35.617181
2	2	郭靖	2301460	30	guojing@csdn.net	2019-12-18 03:51:35.625389
3	3	杨过	4986722	25	yanguo@csdn.net	2019-12-18 03:51:35.627392
4	4	张无忌	1087875	20	zhangwuji@csdn.net	2019-12-18 03:51:35.631379
5	5	王重阳	1650187	60	wangchongyang@csdn.net	2019-12-18 03:51:35.633391
6	6	逍遥子	4486387	100	xiaoyaozi@csdn.net	2019-12-18 03:51:35.636379
7	7	张三丰	4340150	95	zhangsanfeng@csdn.net	2019-12-18 03:51:35.638393
8	8	独孤求败	7883464	30	duguqiubai@csdn.net	2019-12-18 03:51:35.641382
9	9	东方不败	8550976	25	dongfangbubai@csdn.net	2019-12-18 03:51:35.644393
10	10	扫地僧	6110380	66	saodiseng@csdn.net	2019-12-18 03:51:35.646397

4.1.2 开发index_handler

思路： 1. 查找user表中所有的用户数据 2. 把这些数据渲染到index.html中

```
1 from django.shortcuts import render,HttpResponse
2 from user.models import User
3
4 # Create your views here.
5 def index_handler(request):
6     # 查找到所有的user对象
7     users = User.objects.all()
8     # 渲染模板，把users传递过去
9     return render(request,'index.html',context={'users':users})
```



```

1  from django.shortcuts import render,HttpResponse
2  from user.models import User
3
4  # Create your views here.
5  def index_handler(request):
6      # 查找到所有的user对象
7      users = User.objects.all()
8      # 渲染模板,把users传递过去
9      return render(request,'index.html',context={'users':users})
10

```

4.1.3 开发index模板

思路： - 对传递过来的users数据进行遍历，每一个user对象为table中的一行

```

7  <div class="container">
8      <table class="table table-hover">
9          <!-- 表头 -->
10         <thead...>
21         <!-- 表内容 -->
22         <tbody>
23             {% for user in users %}
24                 <tr>
25                     <td>{{ user.id }}</td>
26                     <td>{{ user.username }}</td>
27                     <td>{{ user.password }}</td>
28                     <td>{{ user.age }}</td>
29                     <td>{{ user.email }}</td>
30                     <td>{{ user.createdAt|date:'Y-m-d H:i:s' }}</td>
31                     <td>
32                         <a href="{% url 'alter' user.id %}">修改</a>
33                         <a href="{% url 'del' user.id %}">删除</a>
34                     </td>
35                 </tr>
36             {% endfor %}
37         </tbody>
38     </table>
39 </div>

```

注： - 日期要进行格式化输出 - 在“操作”列中，有两个链接，一个是修改，一个是删除，这两个链接需要对应Id，所以反向解析路由的时候传递每一行数据的Id值 完整代码：`python {#继承base.html#} {% extends 'base.html' %}

{% block title %} 首页 {% endblock %} {#数据展示内容#} {% block article %}

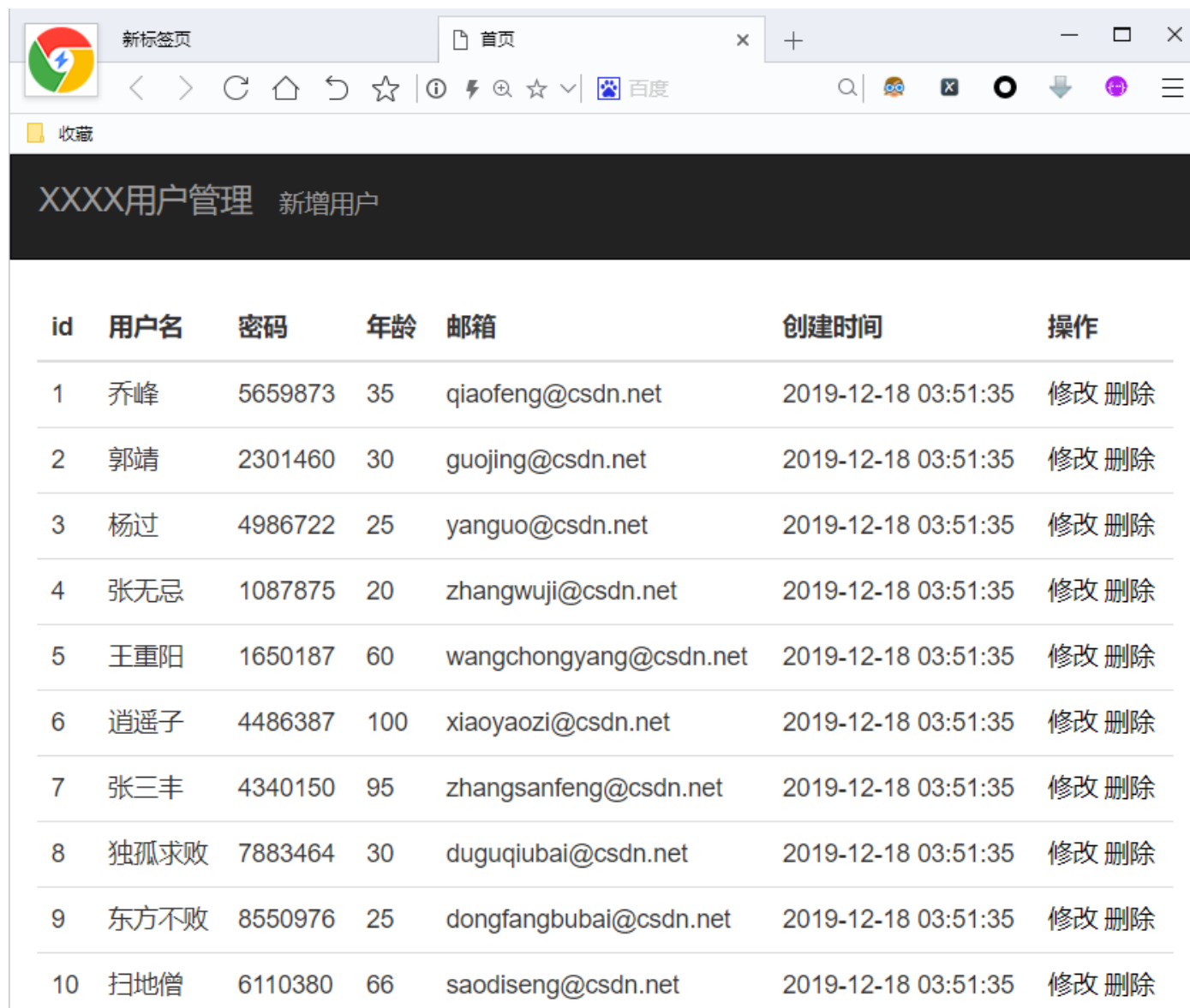
{% for user in users %} {% endfor %}

id	用户名	密码	年龄	邮箱	创建时间	操作
----	-----	----	----	----	------	----

{{ user.id }}	{{ user.username }}	{{ user.password }}	{{ user.age }}	{{ user.email }}	{{ user.createDatetime date:'Y-m-d H:i:s' }}	修改 删除
---------------	---------------------	---------------------	----------------	------------------	----------------------------------------------	----------

{% endblock %}`

4.1.4 测试首页



id	用户名	密码	年龄	邮箱	创建时间	操作
1	乔峰	5659873	35	qiaofeng@csdn.net	2019-12-18 03:51:35	修改 删除
2	郭靖	2301460	30	guojing@csdn.net	2019-12-18 03:51:35	修改 删除
3	杨过	4986722	25	yanguo@csdn.net	2019-12-18 03:51:35	修改 删除
4	张无忌	1087875	20	zhangwuji@csdn.net	2019-12-18 03:51:35	修改 删除
5	王重阳	1650187	60	wangchongyang@csdn.net	2019-12-18 03:51:35	修改 删除
6	逍遥子	4486387	100	xiaoyaozi@csdn.net	2019-12-18 03:51:35	修改 删除
7	张三丰	4340150	95	zhangsanfeng@csdn.net	2019-12-18 03:51:35	修改 删除
8	独孤求败	7883464	30	duguqiubai@csdn.net	2019-12-18 03:51:35	修改 删除
9	东方不败	8550976	25	dongfangbubai@csdn.net	2019-12-18 03:51:35	修改 删除
10	扫地僧	6110380	66	saodiseng@csdn.net	2019-12-18 03:51:35	修改 删除

4.2 新增用户

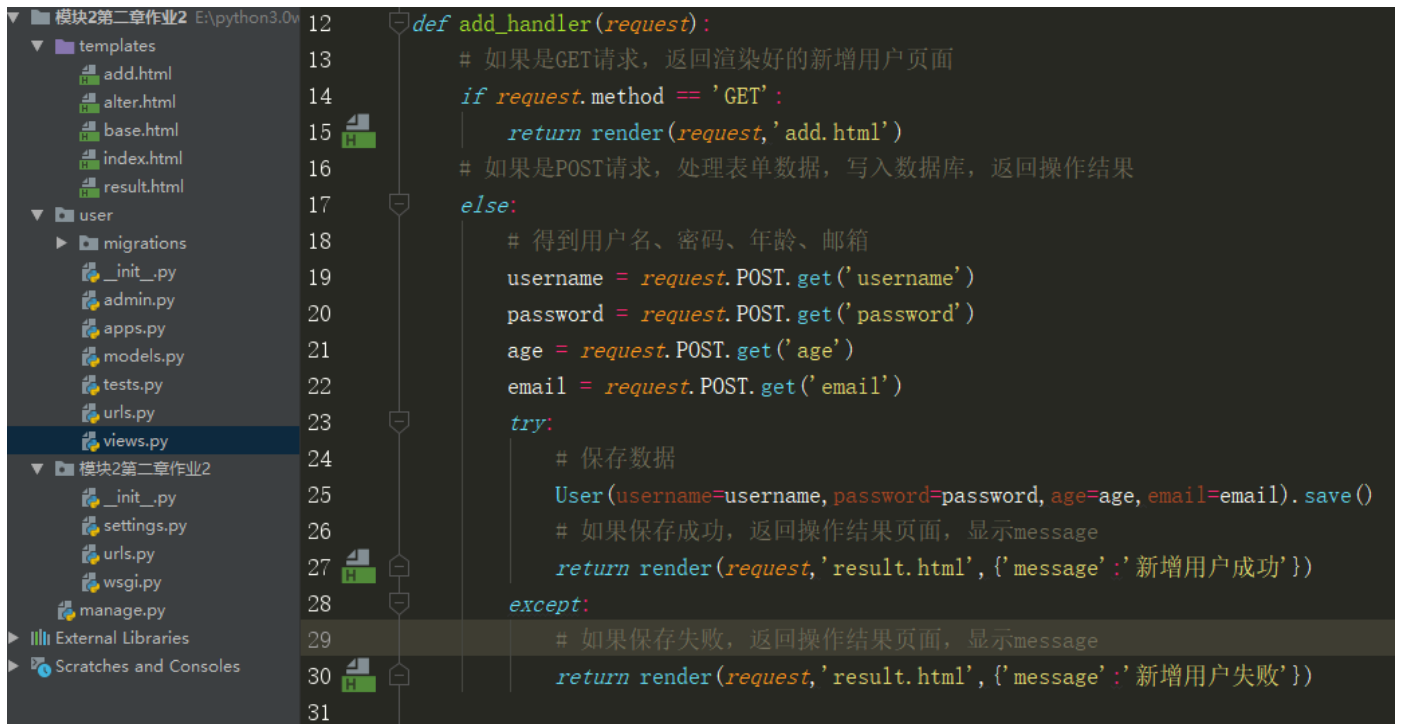
4.2.1 开发add_handler

思路： - 得到用户提交的表单数据 - 把表单数据写入数据库

会遇到的问题： - 包含用户密码的数据，应该采用Post方式提交，如何接收？ - 对于不同的请求方式应该如何处理

Post请求的提交与接收 1. 在前端的Form中需要加入{% csrf_token %} 2. 在后端中，通过username = request.POST.get('username')，得到表单提交的数据

对于不同的请求： 1. Get请求，返回渲染好的add.html模板 2. Post请求，处理提交的新增用户表单数据，写入数据库



```
12 def add_handler(request):
13     # 如果是GET请求，返回渲染好的新增用户页面
14     if request.method == 'GET':
15         return render(request, 'add.html')
16     # 如果是POST请求，处理表单数据，写入数据库，返回操作结果
17     else:
18         # 得到用户名、密码、年龄、邮箱
19         username = request.POST.get('username')
20         password = request.POST.get('password')
21         age = request.POST.get('age')
22         email = request.POST.get('email')
23         try:
24             # 保存数据
25             User(username=username, password=password, age=age, email=email).save()
26             # 如果保存成功，返回操作结果页面，显示message
27             return render(request, 'result.html', {'message': '新增用户成功'})
28         except:
29             # 如果保存失败，返回操作结果页面，显示message
30             return render(request, 'result.html', {'message': '新增用户失败'})
31
```

由于写入数据可能出现异常，所以使用try和except，对于Post请求提交的表单数据处理，不管是否出现异常，都返回渲染后的操作结果页面，但传递不同的message，在result.py中显示message，即操作的结果描述 完整代码： python def add_handler(request): # 如果是GET请求，返回渲染好的新增用户页面 if request.method == 'GET': return render(request, 'add.html') # 如果是POST请求，处理表单数据，写入数据库，返回操作结果 else: # 得到用户名、密码、年龄、邮箱 username = request.POST.get('username') password = request.POST.get('password') age = request.POST.get('age') email = request.POST.get('email') try: # 保存数据 User(username=username,password=password,age=age,email=email).save() # 如果保存成功，返回操作结果页面，显示message return render(request, 'result.html', {'message': '新增用户成功'}) except: # 如果保存失败，返回操作结果页面，显示message return render(request, 'result.html', {'message': '新增用户失败'})

4.2.2 开发add模板

- 1. 添加Form表单的地址action与请求方式method
- 2. 对于Form表单提交的Post请求数据，需要加入{% csrf_token %}



```
1 {% extends 'base.html' %}
2
3 {% block title %} 新增用户 {% endblock %}
4
5 {% block article %}
6     <!-- 表单内容 -->
7     <div class="col-sm-offset-3 col-sm-6">
8         {# 1. 填写Form表单的提交地址action与请求方式method #}
9         <form class="form-horizontal" role="form" action="{% url 'add' %}" method="post">
10             {# 2. 加入csrf_token #}
11             {% csrf_token %}
12             <div class="form-group">
13                 <label for="username" class="col-sm-2 control-label">用户名</label>
14                 <div class="col-sm-10">
15                     <input type="text" class="form-control" id="username" placeholder="请输入用户名" name="username">
16                 </div>
17             </div>
18             <div class="form-group">
```

完整代码：`python {% extends 'base.html' %}

{% block title %} 新增用户 {% endblock %}

{% block article %}

{# 1.填写Form表单的提交地址action与请求方式method #}

{# 2.加入csrf_token #} {% csrf_token %}

用户名

请输入用户名

密码

请输入密码

年龄

请输入年龄

邮箱

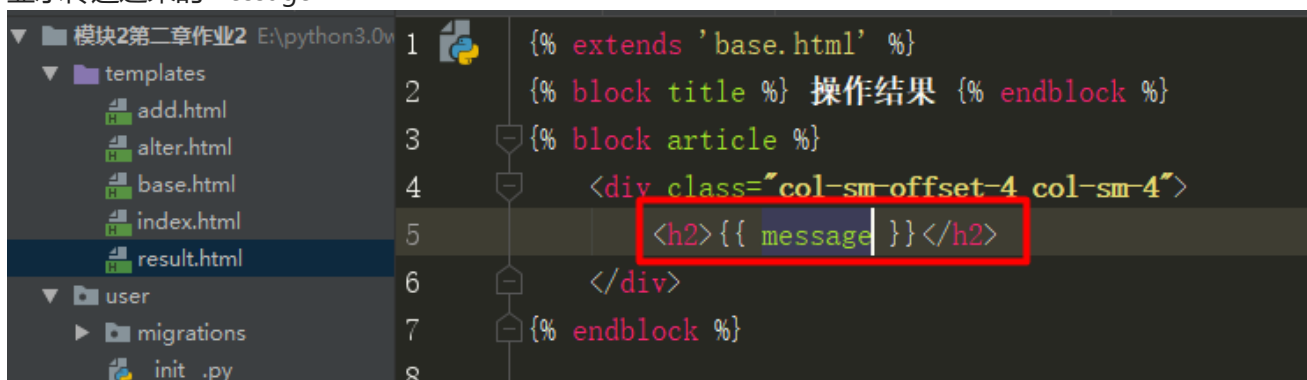
请输入邮箱

添加

{% endblock %}

4.2.3 开发result模板

显示传递过来的message

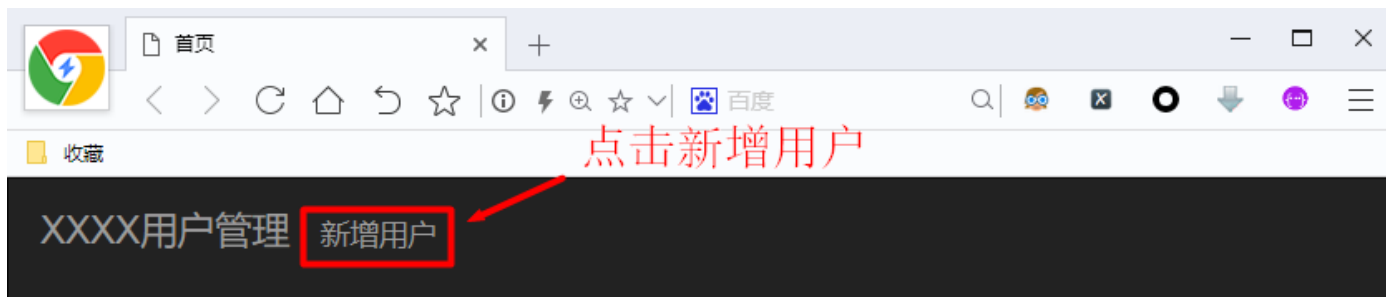


```
1 {% extends 'base.html' %}
2
3 {% block title %} 操作结果 {% endblock %}
4
5 {% block article %}
6     <div class="col-sm-offset-4 col-sm-4">
7         <h2>{{ message }}</h2>
8     </div>
9 {% endblock %}
```

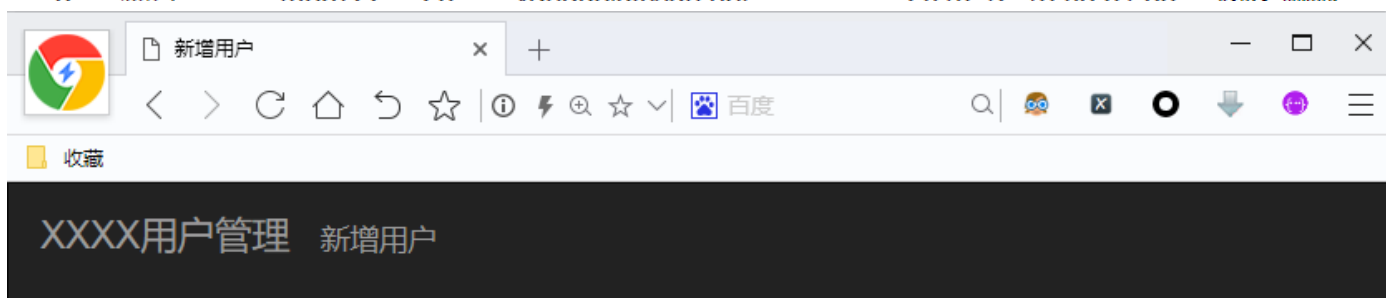
完整

代码：`python {% extends 'base.html' %} {% block title %} 操作结果 {% endblock %} {% block article %} <div class="col-sm-offset-4 col-sm-4"> <h2>{{ message }}</h2> </div> {% endblock %}

4.2.4 测试添加用户



id	用户名	密码	年龄	邮箱	创建时间	操作
1	乔峰	5659873	35	qiaofeng@csdn.net	2019-12-18 03:51:35	修改 删除
2	郭靖	2301460	30	guojing@csdn.net	2019-12-18 03:51:35	修改 删除
3	杨过	4986722	25	yangguo@csdn.net	2019-12-18 03:51:35	修改 删除



用户名

小龙女

1. 数据新增用户数据

密码

123456

2. 点击添加

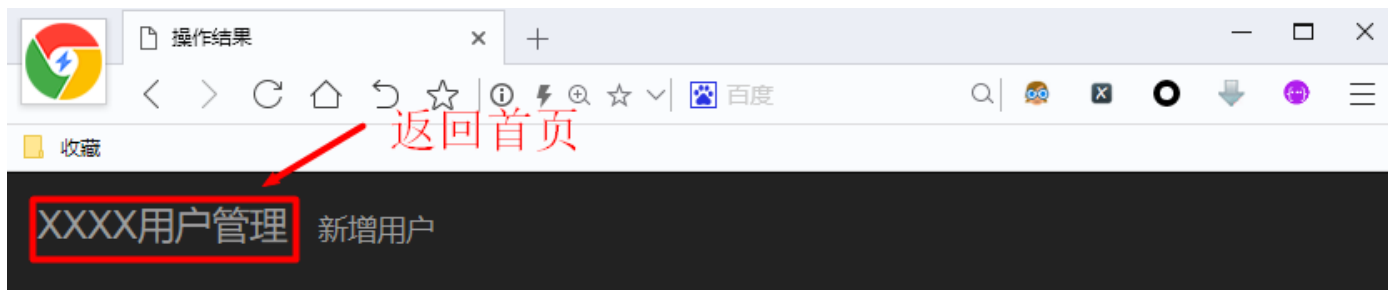
年龄

18

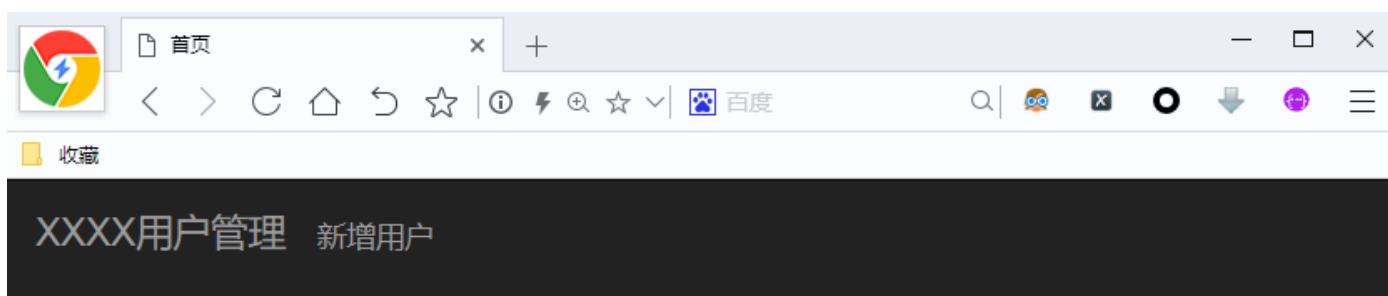
邮箱

xiaolongnv@csdn.net

添加



新增用户成功



id	用户名	密码	年龄	邮箱	创建时间	操作
1	乔峰	5659873	35	qiaofeng@csdn.net	2019-12-18 03:51:35	修改 删除
2	郭靖	2301460	30	guojing@csdn.net	2019-12-18 03:51:35	修改 删除
3	杨过	4986722	25	yanguo@csdn.net	2019-12-18 03:51:35	修改 删除
4	张无忌	1087875	20	zhangwuji@csdn.net	2019-12-18 03:51:35	修改 删除
5	王重阳	1650187	60	wangchongyang@csdn.net	2019-12-18 03:51:35	修改 删除
6	逍遥子	4486387	100	xiaoyaozi@csdn.net	2019-12-18 03:51:35	修改 删除
7	张三丰	4340150	95	zhangsanfeng@csdn.net	2019-12-18 03:51:35	修改 删除
8	独孤求败	7883464	30	duguqiubai@csdn.net	2019-12-18 03:51:35	修改 删除
9	东方不败	8550976	25	dongfangbubai@csdn.net	2019-12-18 03:51:35	修改 删除
10	扫地僧	6110380	66	saodiseng@csdn.net	2019-12-18 03:51:35	修改 删除
11	小龙女	123456	18	xiaolongnv@csdn.net	2019-12-18 05:26:41	修改 删除

4.3 修改用户

4.3.1 开发alter_handler

思路： - 如果是Get请求： - 返回修改用户页面，并做到数据填充（即不是空的表单对象，方便用户修改） - 如果是Post请求 - 根据路由中传递的id，锁定要修改的用户对象 - 得到表单提交的数据，完成修改 - 返回操作结果

完整代码：python def alter_handler(request,id): # 得到被修改的user对象 user = User.objects.get(id=id) # 如果是GET请求，渲染页面 if request.method == 'GET': return render(request,'alter.html',context={'user':user}) # 如果是POST请求，处理数据 else: # 得到用户名、密码、年龄、邮箱 user.username = request.POST.get('username') user.password = request.POST.get('password') user.age = request.POST.get('age') user.email = request.POST.get('email') # 如果修改成功 try: user.save() return render(request,'result.html',context={'message':'修改用户成功'}) # 如果修改不成功 except: return render(request,'result.html',context={'message':'修改用户失败'})

4.3.2 开发alter模板


```
4 {% block article %}
5 <!-- 表单内容 -->
6 <div class="col-sm-offset-3 col-sm-6">
7   {# 1. 修改action与方法 #}
8   <form class="form-horizontal" role="form" action="{% url 'alter' user.id %}" method="post">
9     {# 2. 加入csrf_token, 执行Post请求 #}
10    {% csrf_token %}
11    <div class="form-group">
12      <label for="username" class="col-sm-2 control-label">用户名</label>
13      <div class="col-sm-10">
14        <input type="text" class="form-control" id="username" placeholder="请输入用户名" name="username" value="{{ user.username }}">
15      </div>
16    </div>
17    <div class="form-group">
18      <label for="password" class="col-sm-2 control-label">密码</label>
19      <div class="col-sm-10">
20        <input type="text" class="form-control" id="password" placeholder="请输入密码" name="password" value="{{ user.password }}">
21      </div>
22    </div>
23    <div class="form-group">
24      <label for="age" class="col-sm-2 control-label">年龄</label>
25      <div class="col-sm-10">
26        <input type="text" class="form-control" id="age" placeholder="请输入年龄" name="age" value="{{ user.age }}">
27      </div>
28    </div>
29    <div class="form-group">
30      <label for="email" class="col-sm-2 control-label">邮箱</label>
31      <div class="col-sm-10">
32        <input type="text" class="form-control" id="email" placeholder="请输入邮箱" name="email" value="{{ user.email }}">
33      </div>
34    </div>
```

这里填充默认值，为了方便用户操作 完整代码：`python {% extends 'base.html' %}

{% block title %} 修改用户 {% endblock %}

{% block article %}

{# 1.修改action与方法 #}

{# 2.加入csrf_token, 执行Post请求 #} {% csrf_token %}

用户名

{{ user.username }}

密码

{{ user.password }}

年龄

{{ user.age }}

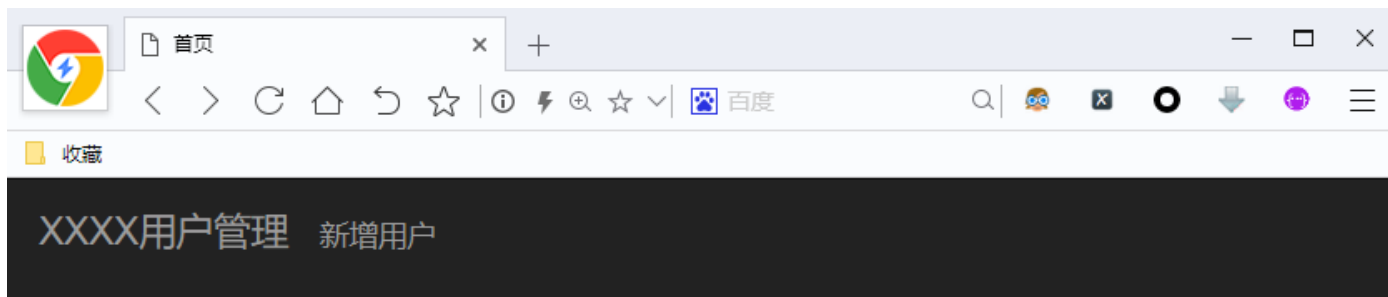
邮箱

{{ user.email }}

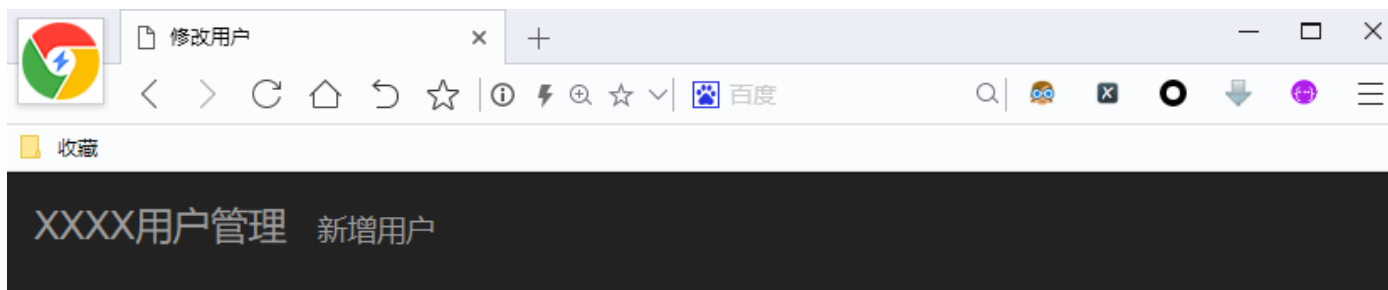
修改

{% endblock %}

4.3.3 测试修改用户



id	用户名	密码	年龄	邮箱	创建时间	操作
1	乔峰	5659873	35	qiaofeng@csdn.net	2019-12-18 03:51:35	修改 删除
2	郭靖	2301460	30	guojing@csdn.net	2019-12-18 03:51:35	修改 删除



用户名

乔峰

密码

5659873

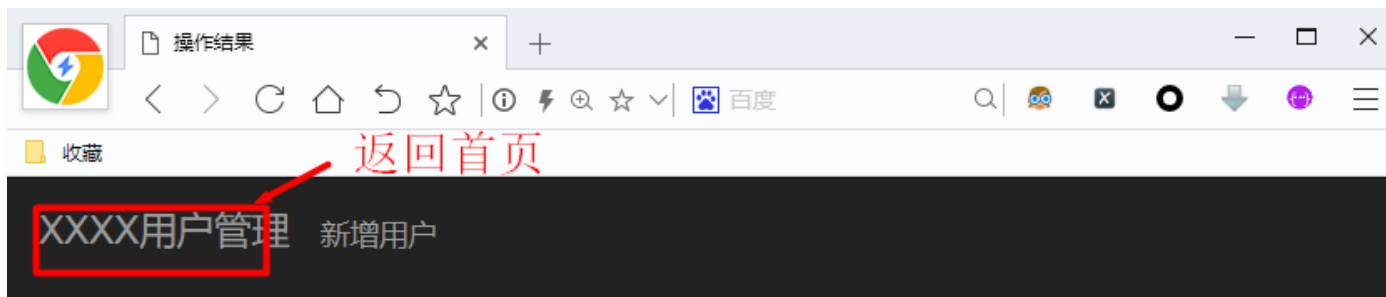
年龄

40

邮箱

qiaofeng@csdn.net

修改

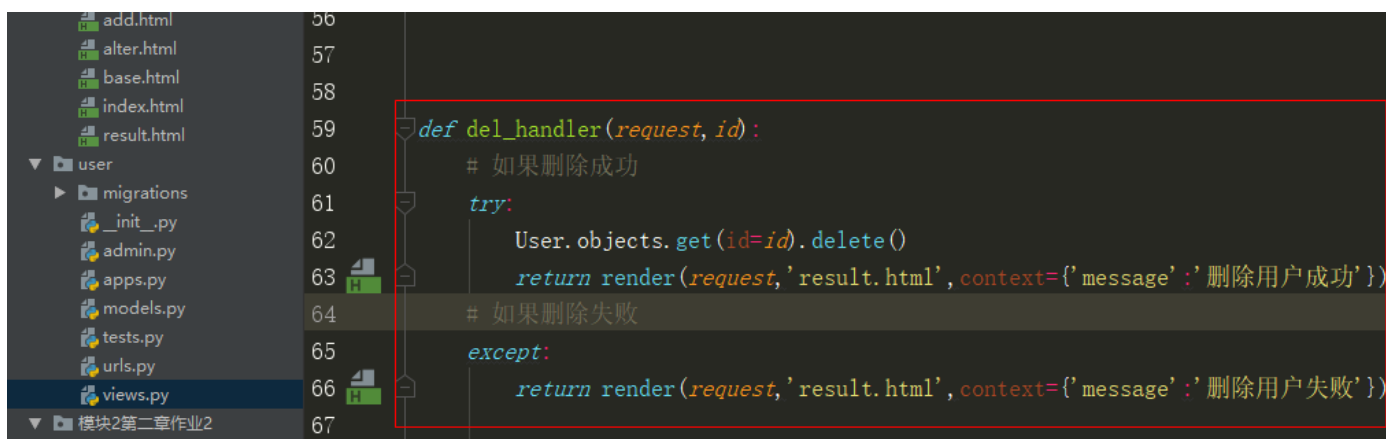


修改用户成功

id	用户名	密码	年龄	邮箱	创建时间	操作
1	乔峰	5659873	40	qiaofeng@csdn.net	2019-12-18 03:51:35	修改 删除
2	郭靖	2301460	30	guojing@csdn.net	2019-12-18 03:51:35	修改 删除
3	杨过	4986722	25	yanguo@csdn.net	2019-12-18 03:51:35	修改 删除

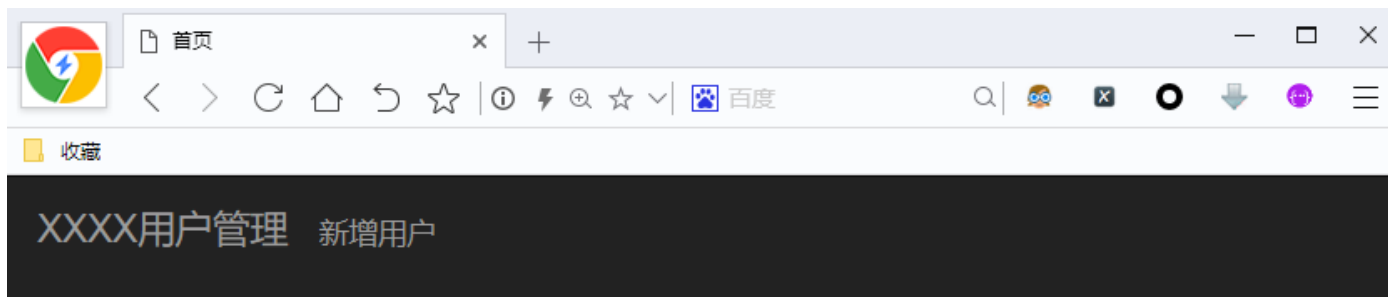
4.4 删除用户

4.4.1 开发del_handler

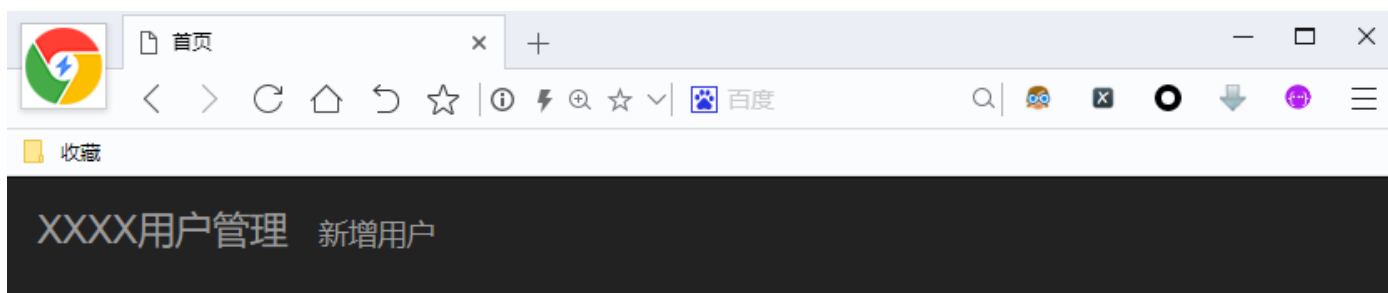


完整代码: `python def del_handler(request,id): # 如果删除成功 try: User.objects.get(id=id).delete() return render(request,'result.html',context={'message':'删除用户成功'}) # 如果删除失败 except: return render(request,'result.html',context={'message':'删除用户失败'})`

4.4.2 测试删除用户



删除用户成功



5 评分标准

1. 构造模型层并生成对应的表结构 10分
2. 完成增删改查功能的开发 20分
3. 代码注释，规范10分