# 还是day16项目,我们上一节实现了,视图分类管理,公共功能抽取等等的功能使得我们的项目已经比较由条理了,具体可以参考: [django学习笔记9](django学习笔记9-用django开发员工管理系统3%20-%20ModelForm3.docx)

# 这节课我们来实现用户登录和权限相关功能的开发

# 1.实现管理员相关的功能

## 1>创建管理员表,这张表笔记简单只有3列

|  |
| --- |
|  |

### 我们在models.py里面添加一个Admin类

|  |
| --- |
|  |

### 然后我们需要使用manage.py来生成对应的表

|  |
| --- |
|  |
|  |
|  |
|  |

## 2>在layout.html里面的导航条上面添加一个链接跳转到管理员账户管理功能

|  |
| --- |
|  |

## 3>然后我们在urls.py里面添加一个路由

|  |
| --- |
|  |

### 然后我们在views文件夹里面创建一个admin\_view.py文件,实现这个函数

|  |
| --- |
|  |

## 4>然后我们需要在templates里面新建一个admin\_list.html模板文件,需要继承自layout,注意在这里是不能够显示用户名的密码的,因为这是管理员权力很大,需要保护好密码.所以我们用\*\*\*\*\*\*\*代替

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>管理员列表</**title**> {% **endblock** %}  {% **block content** %}  <**div class="container"**>  <**div style="margin-bottom**: 5**px**;**" class="clearfix"**>  <**a href="/admin/add/" class="btn btn-primary"**>  <**span class="glyphicon glyphicon-plus-sign" aria-hidden="true"**></**span**>  新增管理员  </**a**>  <**div style="float**: **right**;**width**:300**px**;**"**>  <**form method="get"**>  <**div class="input-group"**>  <**input type="text" name="kw" class="form-control" placeholder="关键字..." value="**{{ **kw** }}**"**>  <**span class="input-group-btn"**>  <**button class="btn btn-default" type="submit"**>  <**span class="glyphicon glyphicon-search" aria-hidden="true"**></**span**>  </**button**>  </**span**>  </**div**>  </**form**>  </**div**>  </**div**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**span class="glyphicon glyphicon-th-list" aria-hidden="true"**></**span**>  管理员列表  </**div**>   <**table class="table table-bordered table-responsive"**>  <**thead**>  <**tr**>  <**th**>ID</**th**>  <**th**>用户名</**th**>  <**th**>密码</**th**>  **<th>**操作**</th>**  </**tr**>  </**thead**>  <**tbody**>  {% **for ad in admins** %}  <**tr**>  <**td**>{{ **ad**.**id** }}</**td**>  <**td**>{{ **ad**.**username** }}</**td**>  <**td**>\*\*\*\*\*\*\*\*\*</**td**>  <**td**> *{# <a href="/dep/edit/?nid={{ dep.id }}" class="btn btn-success btn-xs">编辑</a>#}* <**a href="/admin/**{{ **ad**.**id** }}**/edit/" class="btn btn-success btn-xs"**>编辑</**a**>  <**a href="/admin/**{{ **ad**.**id**}}**/del/" class="btn btn-danger btn-xs"**>删除</**a**>  </**td**>  </**tr**>  {% **endfor** %}  </**tbody**>  </**table**>  </**div**>  *{# 弹性盒子 #}* <**div class="clearfix"**>  *{# 分页组件 #}* <**ul class="pagination"**>  {{**page\_str**}}  </**ul**>   </**div**> </**div**> {% **endblock** %} |

### 为了方便观察,我们先往admin表插入一条数据

|  |
| --- |
|  |

### 然后运行程序,效果如下

|  |
| --- |
|  |

### 然后我们来实现搜索功能,把admin\_list函数修改一下,在前面添加如下代码并且把all方法改为filter方法,因为如果条件为空,filter的效果和all是一样的

|  |
| --- |
|  |

### 效果:

|  |
| --- |
|  |
|  |

## 5>现在我们来实现新建管理员功能.先在urls.py条件一个路由映射,

|  |
| --- |
|  |

### 然后我们在day16forms.py里面新建一个AdminAddForm,从我们的BootstrapModelForm继承,这里需要验证用户名不能够重复

|  |
| --- |
|  |

### 然后在admin\_view里面实现这个admin\_add函数

|  |
| --- |
|  |

### 然后我们需要在templates文件夹里面创建一个admin\_add.html文件,内容如下

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>新增管理员</**title**> {% **endblock** %}   {% **block content** %} <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>新建管理员</**h3**>  </**div**>  <**div class="panel-body"**>  *{# 表单 #}* <**form action="/admin/add/" method="post" novalidate**>  {% **csrf\_token** %}  {% **for item in form** %}  <**div class="form-group"**>  {{ **item**.**label** }}  {{ **item** }}  <**span style="color**: **red**;**"**> {{ **item**.**errors**.0 }}</**span**>  </**div**>  {% **endfor** %}   <**button type="submit" class="btn btn-success"**>保 存</**button**>  </**form**>  </**div**>  </**div**>  </**div**> </**div**> {% **endblock** %} |

### 效果:

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

## 6>我们来完成编辑管理员的功能,先在urls.py里面创建一个路由映射,

|  |
| --- |
|  |

### 然后我们需要在day16forms.py里面新建一个AdminEditForm从BootstrapModelForm继承

|  |
| --- |
|  |

### 然后我们在admin\_view.py文件里面创建一个admin\_edit函数,在里面使用我们的AdminEditForm来完成任务

|  |
| --- |
|  |

### 然后我们创建关于admin\_edit.html文件,在里面渲染form对象的数据

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>编辑管理员</**title**> {% **endblock** %}   {% **block content** %} <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>编辑管理员</**h3**>  </**div**>  <**div class="panel-body"**>  *{# 表单 #}* <**form method="post" novalidate**>  {% **csrf\_token** %}  {% **for item in form** %}  <**div class="form-group"**>  {{ **item**.**label** }}  {{ **item** }}  <**span style="color**: **red**;**"**> {{ **item**.**errors**.0 }}</**span**>  </**div**>  {% **endfor** %}   <**button type="submit" class="btn btn-success"**>保 存</**button**>  </**form**>  </**div**>  </**div**>  </**div**> </**div**> {% **endblock** %} |

### 效果

|  |
| --- |
|  |
|  |
|  |
|  |

## 7>然后我们来实现删除管理员功能,先在urls.py里面添加一个路由映射admin/del/

|  |
| --- |
|  |

### 然后在admin\_view.py文件里面添加admin\_del函数

|  |
| --- |
|  |

### 效果,我们在最后一个管理员上面点击删除,此时就只剩下一页数据了

|  |
| --- |
|  |

### 到数据库中查看,也是只有3条记录了

|  |
| --- |
|  |

# 2.代码优化,抽取相同的代码实现模板代码复用

## 我们前面的user\_add.html,pretty\_add.html和admin\_add.html其实代码都是大同小异,我们完全可以用一个通用模板然后对不同的部分进行设置,这样子就可以做的一个文件多次使用

## 1>我们把day16项目复杂一份,改名day18,只是项目根目录修改了,其他比如它里面的包都是day16的.内容还是一模一样.在确认它能够正常运行后,我们在这里优化代码

|  |
| --- |
|  |
|  |

## 2>把admin\_add.html复制一份,改名add\_new.html,然后我们来修改代码

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>{{ **title** }}</**title**> {% **endblock** %} {% **block content** %}  <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>{{ **title** }}</**h3**>  </**div**>  <**div class="panel-body"**>  *{# 表单 #}* <**form method="post" novalidate**>  {% **csrf\_token** %}  {% **for item in form** %}  <**div class="form-group"**>  {{ **item**.**label** }}  {{ **item** }}  <**span style="color**: **red**;**"**> {{ **item**.**errors**.0 }}</**span**>  </**div**>  {% **endfor** %}   <**button type="submit" class="btn btn-success"**>保 存</**button**>  </**form**>  </**div**>  </**div**>   </**div**> </**div**>  {% **endblock** %} |

## 3>然后我们可以修改pretty\_add函数的代码如下

|  |
| --- |
|  |

### 然后修改admin\_add函数的代码如下

|  |
| --- |
|  |

#### 注意:user\_add函数的代码不能改,因为他用到的user\_add.html由日期选择器功能,这个比较特殊,跟其他页面不通用.必须保留

### 我们可以把pretty\_add.html和admin\_add.html文件删除

## 4>我们发现,其实add\_new.html模板也能够用在比较视图函数里面,所以我们把pretty\_edit.html和admin\_edit.html模板文件删除,然后把add\_new.html文件改名add\_or\_edit.html,这个名字比较由通用性,然后在可以使用它的地方使用.

|  |
| --- |
|  |

## 5>我们可以给新增管理员的页面添加一个再次输入密码,用来核对2次输入的密码是否一致的功能,文件需要在我们的AdminAddForm中创建一个confirm\_pwd字段.注意:数据库中没有这个字段

|  |
| --- |
|  |

### 那么问题来了,我们如何在提交数据的时候检查这两个密码是否一致呢?可以利用form.clean\_data来实现还需要在我们的AdminAddForm类里面使用钩子函数来实现

|  |
| --- |
| **from django.core.exceptions import ValidationError from django.core.validators import RegexValidator  from day16app.utils.basemodelform import BootstrapModelForm from django import forms from day16app import models from day16app.models import UserInfo, Department**  **class** AdminAddForm(BootstrapModelForm):  *# 核对2次输入密码是否一致* confirm\_pwd = forms.CharField(  label=**"确认密码"**,  widget=forms.PasswordInput *# 不显示密码* )   **class** Meta:  model = models.Admin  fields = [**"username"**, **"password"**, **"confirm\_pwd"**]  widgets ={  **"password"**:forms.PasswordInput *# 把密码字段也隐藏* }   *# 管理员账户不能重复,钩子函数来的* **def** clean\_username(self):  txt\_username = self.cleaned\_data[**"username"**]  exist = models.Admin.objects.filter(username=txt\_username).exists()  **if** exist:  **raise** ValidationError(**"用户名不能重复"**) *# 验证失败就抛异常* **return** txt\_username *# 验证通过就把他返回   # 确认密码的钩子* **def** clean\_confirm\_pwd(self):  password = self.cleaned\_data[**"password"**]  confirm\_pwd = self.cleaned\_data[**"confirm\_pwd"**]  **if** confirm\_pwd != password:  **raise** ValidationError(**"确认密码和原来的密码不一样"**) *# 验证失败就抛异常* **return** confirm\_pwd |

### 此时如果你两次输入的密码不一样,就会报错

|  |
| --- |
|  |

### 把两个地方的密码改为一致,就可以顺利添加

|  |
| --- |
|  |

### 注意:这个密码检查不通过的话,默认是会清空密码框的内容,如果你不想它清空,你需要给forms.PasswordInput函数传递一个render\_value=True

|  |
| --- |
|  |

### 我是喜欢他清空的.

### 6>还有一个问题,就是密码其实需要进行md5加密后才能够存储到数据库