# 还是员工管理系统,前面的代码请参考: [django开发笔记y7](django学习笔记7-用django开发员工管理系统%20-%20模板继承+ModelForm.docx)

|  |
| --- |
|  |

# 1.这里我们继续来完成用户的编辑和删除功能

## 1>在user\_list.html文件里面的编辑链接里面添加路由/user/{{user.id}}/edit

|  |
| --- |
|  |

## 2>然后我们需要在urls.py文件里面新建员工路由匹配,把user/<int:nid>/edit/ 和views.py里面的user\_edit函数对应起来

|  |
| --- |
|  |

## 3>然后我们需要在views.py里面编写user\_edit函数

|  |
| --- |
|  |

## 4> 我们在templates文件夹里面新建一个user\_edit.html文件 ,这里需要用到一个技巧,这里的form里面不能够写action,因为你不能够获取到,你不写,它默认会提交到当前链接,所以是没有问题的.

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>编辑用户</**title**> {% **endblock** %} {% **block content** %}  <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>编辑用户</**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** %} |

### 此时点击用户列表里面的编辑按钮,就会跳转到编辑页面

|  |
| --- |
|  |

### 不过此时表单中没有填充数据,那么ModelForm如何填充数据呢?

## 5>我们在views.py的user\_edit函数里面根据nid查找到一个用户,然后在创建form实例的时候把它作为instance参数传递给MyFrom的构造函数即可

|  |
| --- |
|  |

### 然后我们就可以在页面中看到数据了

|  |
| --- |
|  |

## 6>然后我们来进行表单验证,其实这一部分和添加用户的代码是类似的,只需要复制下来修改即可

|  |
| --- |
|  |

### 然后我们进入编辑页面,故意把一些字段留空,发现验证起作用了

|  |
| --- |
|  |
|  |

### 然后我们输入正确的数据,点击保存,

|  |
| --- |
|  |

### 发现数据修改了

|  |
| --- |
|  |

## 7>其实我们是可以获取到提交地址的,因为views.user\_edit函数在处理get请求的时候就得到了一个nid我们可以把它传递给模板

|  |
| --- |
|  |
|  |

## 8>其实views.user\_edit函数的代码可以优化一下

|  |
| --- |
|  |

## 9.我们是项目还有一个小问题,就是绕转时间其实只需要年月日就可以了,我们需要来修改一下模型

|  |
| --- |
|  |

## 10.然后需要重新进行数据库迁移

|  |
| --- |
|  |
|  |

### 注意,此时在user\_list.html里面我们需要把日期格式的H:i:s去掉,否则程序会崩溃

|  |
| --- |
|  |

### 然后用户列表就是这样子的

|  |
| --- |
|  |

## 11.如果ModelForm出来保存用户输入的值,还需要保存其他值,可以这么写:form.instance.字段名=值

|  |
| --- |
|  |

## 13.然后我们来实现删除功能,我们先把user\_list.html里面的删除链接的url改为user/{{user.id}}/del/

|  |
| --- |
|  |

### 然后我们在urls.py里面添加一个user/<int:nid>/del/路由,把它和views.py里面的user\_del函数对应起来

|  |
| --- |
|  |

### 测试一下,比如我们输出id为8的一行数据

|  |
| --- |
|  |

## 发现数据被删除了

|  |
| --- |
|  |

# 2.然后我们添加一个靓号管理模块

## 1>这个模块的数据库表结构要求如下

|  |
| --- |
|  |

## 2>我们需要在models.py在定义一个PrettyNumber类

|  |
| --- |
|  |

## 3>然后我们需要利用manage.py来生成数据库表

|  |
| --- |
|  |
|  |

## 4>为了方便学习,我们需要在数据库中手动输入一些数据

|  |
| --- |
|  |

## 5>我们来做第一个功能靓号列表,首先需要在urls.py里面定义一个pretty/list/路由,与views.py里面的pretty\_list函数对应

|  |
| --- |
|  |

## 6>然后我们在views.py在创建一个pretty\_list函数,注意需要先把这个类导入.我们还是导入models就好了,然后使用models.PrettyNuber来操作数据

|  |
| --- |
|  |

### 然后我们需要获取prettynumber表的所以数据,然后渲染一个pretty\_list.html文件,把它数据传递进去

|  |
| --- |
|  |

## 7>在templates文件夹里面新建一个pretty\_list.html,继承layout.html

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %} <**title**>靓号列表</**title**> {% **endblock** %}  {% **block content** %} <**div class="container"**> <**div style="margin-bottom**: 5**px**;**"**>  <**a href="/pretty/add/" class="btn btn-primary"**>  <**span class="glyphicon glyphicon-plus-sign" aria-hidden="true"**></**span**>  新增靓号  </**a**>  </**div**> <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**span class="glyphicon glyphicon-th-list" aria-hidden="true"**></**span**>  靓号列表  </**div**>   <**table class="table table-bordered table-responsive"**>  <**thead**>  <**tr**>  <**th**>ID</**th**>  <**th**>手机号</**th**>  <**th**>价格</**th**>  <**th**>级别</**th**>  <**th**>状态</**th**>  </**tr**>  </**thead**>  <**tbody**>  {% **for pr in pretties** %}  <**tr**>  <**td**>{{ **pr**.**id** }}</**td**>  <**td**>{{ **pr**.**mobile** }}</**td**>  <**td**>{{**pr**.**price**}}</**td**>  <**td**>{{ **pr**.**level** }}</**td**>  <**td**>{{ **pr**.**status** }}</**td**>  <**td**> *{# <a href="/dep/edit/?nid={{ dep.id }}" class="btn btn-success btn-xs">编辑</a>#}* <**a href="/pretty/**{{ **pr**.**id** }}**/edit/" class="btn btn-success btn-xs"**>编辑</**a**>  <**a href="/pretty/**{{ **pr**.**id**}}**/del/" class="btn btn-danger btn-xs"**>删除</**a**>  </**td**>  </**tr**>  {% **endfor** %}  </**tbody**>  </**table**>  </**div**> </**div**> {% **endblock** %} |

## 8>然后我们需要修改一些layout.html,添加一个管理靓号的链接,这样子我们就不用在浏览器手动输入地址.

|  |
| --- |
|  |

### 测试一下,重启服务器,进入首页,然后点击靓号管理

|  |
| --- |
|  |

### 然后就会进入靓号列表页面,可以看到我们添加的数据

|  |
| --- |
|  |

### 注意:这样子是修改不好,级别和状态我们需要用文字显示,所以我们需要修改一下pretty\_display.html,level用get\_level\_display代替,status用get\_status\_display代替

|  |
| --- |
|  |

### 然后这个才是我们需要的效果

|  |
| --- |
|  |

### 注意我们的数据有点小问题,老师的标准是数字越大,级别越高,我们把它修改一下

|  |
| --- |
|  |

### 然后我们在获取数据的时候对它进行排序,使用order\_by()方法,如果传递id表示按id升序排列,如果传递-id表示按id降序排列

|  |
| --- |
|  |

### 效果如下

|  |
| --- |
|  |

## 9>添加新增靓号功能,先在urls.py里面添加一个pretty/add/路由,然后把他和views.pretty\_add函数对应起来

|  |
| --- |
|  |

### 然后我们在views.py文件里面添加一个pretty\_add函数,在这里需要先定义一个表单类继承ModelForm然后把他和PrettyNumber数据模型类关联起来

|  |
| --- |
|  |

### 注意fields的新写法,它还可以指定排除哪些字段

|  |
| --- |
|  |

### 然后再来写pretty\_add函数的代码

|  |
| --- |
|  |

### 然后我们来测试添加数据

|  |
| --- |
|  |

### 点击保存后靓号添加成功

|  |
| --- |
|  |

### 为了统一风格,我把MyForm改名为UserInfoForm

|  |
| --- |
|  |

## 10.然后我们来实现编辑靓号的功能,先在urls.py里面添加一个pretty/<int:nid>/edit/路由,对应views.py里面的pretty\_edit函数

|  |
| --- |
|  |

### 然后我们在views.py里面添加一个pretty\_edit函数.在里面渲染一个叫做pretty\_edit.html模板,我们需要在get路由里面获取nid,然后找到id=nid的一行数据把他传递给模板,

|  |
| --- |
|  |

### 然后我们需要在templates文件夹里面新建一个pretty\_edit.html模板文件

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>修改靓号</**title**> {% **endblock** %} {% **block content** %}  <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>修改靓号</**h3**>  </**div**>  <**div class="panel-body"**>  *{# 表单 不需要写acttion,这里获取不到,它会提交到当前路由 #}* <**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** %} |

### 然后我们在靓号列表里面下载一个号码来修改

|  |
| --- |
|  |

### 点击保存

|  |
| --- |
|  |

### 发现数据已经成功修改

|  |
| --- |
|  |

### 这里,需要把表单验证做好一点因为手机号码编写是数字字符串,我们需要使用正则表达式来验证一个号码是否是手机号码.我们需要重写手机号码字段

|  |
| --- |
|  |

### 这样子,如果用户在手机一栏输入非数字字符串,就会报错

|  |
| --- |
|  |

### 添加验证器还有第二种方式,就是添加一个函数来实现,这个函数的名称必须是clean\_字段名称如:clean\_mobile

|  |
| --- |
|  |

### 注意,验证器有一个副作用,就是会导致/pretty/<int:nid>/edit/中的nid变为空,这样子就会得到错误的路径/pretty//edit/

### 这个函数是一个钩子方法,他是根据你输入的字段来帮你生成的clean\_字段名称钩子函数

### 另外,如果你需要编辑页面里面有某个字段不可编辑,你需要重写定义这个字段

|  |
| --- |
|  |

### 还有一个问题,就是手机号不能重复,这个这么办?

## 11.解决手机号不能重复的问题,我们需要让他在提交数据到数据库之前先去数据库查找一下指定的手机号是否存在可以使用orm的exist函数来判断手机号是否已经存在

|  |
| --- |
|  |

### 我们需要在钩子函数里面修改代码,用正则表达式来验证手机号格式,然后用钩子函数来判断手机号码是否重复

|  |
| --- |
|  |

### 我们测试一些,点击新增手机号,然后填写一个已经存在的号码点击保存

|  |
| --- |
|  |
|  |

## 12然后需要注意,我们在编辑的时候是可以使用同一个手机号码的,此时如果两个路由都使用同一个form对象,就会发现不能够编辑信息了.所以这里我们需要针对pretty\_add函数添加一个form类,针对pretty\_edit函数添加另外一个form类(这个类里面不允许编辑手机号.),也就是使用不同的类,代码有所不同.

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

### 此时添加页面就会有检测手机号重复的功能

|  |
| --- |
|  |

### 编辑页面的手机号一栏被禁用

|  |
| --- |
|  |

### 如果你需要他允许编辑,你需要添加一个条件,就是排除了自己以外,手机号不能重复.

|  |
| --- |
|  |

### 我们来练习一下这种方式

|  |
| --- |
|  |

### 此时如果我们把当前手机号改为已经存在的手机号,就会报错

|  |
| --- |
|  |
|  |

### 但是,如果我不修改手机号,改其他信息

|  |
| --- |
|  |

### 就可以修改成功

|  |
| --- |
|  |

## 13.然后我们来实现生成靓号的操作首先,需要在pretty\_list.html里面的删除超链接添加路由

|  |
| --- |
|  |

### 然后我们需要在urls.py里面把pretty/<int:nid>/del/路由和views.py里面的pretty\_del函数对应起来

|  |
| --- |
|  |

### 然后我们在views.py里面编写代码

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

### 删除成功.