# 还是上面的员工管理项目,参考出口:[django学习笔记6-开发员工管理系统](django学习笔记6-用django开发员工管理系统.docx)

# 我们在上一节中有些模板文件其实内容是几乎一样的,我们有重新写了一遍,这样子的效率不高,django为我们提供了模板继承机制,可以实现模板文件代码复用.我们可以把代码的公共部分抽取出来做员工父模板,然后在某一个子模板里面继承这个父模板.

# 1.我们可以对day16项目的模板做一些改进

## 1.在templates文件夹里面创建员工layout.html文件,把dep\_list.html文件的内容拷贝过去,然后保留css和导航条的内容,把下面的列表内容删除只保留1个div

|  |
| --- |
| {% **load static** %} <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>    <**link rel="stylesheet" href="**{% **static 'plugins/bootstrap-3.4.1/css/bootstrap.min.css'** %}**"**>  <**style**>  .**table**{  **width**: 100%;  }  .**navbar**{  **border-radius**: 0;  }  </**style**> </**head**> <**body**>  <**nav class="navbar navbar-default"**>  <**div class="container"**>  *<!-- Brand and toggle get grouped for better mobile display -->* <**div class="navbar-header"**>  <**button type="button" class="navbar-toggle collapsed" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1" aria-expanded="false"**>  <**span class="sr-only"**>Toggle navigation</**span**>  <**span class="icon-bar"**></**span**>  <**span class="icon-bar"**></**span**>  <**span class="icon-bar"**></**span**>  </**button**>  <**a class="navbar-brand" href="#"**>用户管理系统</**a**>  </**div**>   *<!-- Collect the nav links, forms, and other content for toggling -->* <**div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1"**>  <**ul class="nav navbar-nav"**>  <**li**><**a href="/dep/list"**>部门管理</**a**></**li**>  <**li**><**a href="#"**>Link</**a**></**li**>  </**ul**>  <**ul class="nav navbar-nav navbar-right"**>  <**li**><**a href="#"**>登录</**a**></**li**>  <**li class="dropdown"**>  <**a href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false"**>当前用户名称 <**span class="caret"**></**span**></**a**>  <**ul class="dropdown-menu"**>  <**li**><**a href="#"**>个人资料</**a**></**li**>  <**li**><**a href="#"**>我的信息</**a**></**li**>  <**li role="separator" class="divider"**></**li**>  <**li**><**a href="#"**>注销</**a**></**li**>  </**ul**>  </**li**>  </**ul**>  </**div**>*<!-- /.navbar-collapse -->* </**div**>*<!-- /.container-fluid -->* </**nav**>  <**div**>  </**div**>  <**script src="**{% **static 'js/jquery3.7.1.min.js'** %}**"**></**script**>  <**script src="**{% **static 'plugins/bootstrap-3.4.1/js/bootstrap.min.js'** %}**"**></**script**> </**body**> </**html**> |

## 2.然后在在title并且的位置放置一个{% **block title** %}{% **endblock** %}占位符,在需要放置内容的div中放员一个位符{% **block content** %} {% **endblock** %}

|  |
| --- |
|  |
|  |

## 3.然后我们来改造dep\_list.html,把它变为下面的样子

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %} <**title**>部门列表</**title**> {% **endblock** %}  {% **block content** %} <**div class="container"**> <**div style="margin-bottom**: 5**px**;**"**>  <**a href="/dep/add/" class="btn btn-primary"**>  <**span class="glyphicon glyphicon-plus-sign" aria-hidden="true"**></**span**>  新增部门  </**a**>  </**div**> <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**span class="glyphicon glyphicon-th-list" aria-hidden="true"**></**span**>  部门列表  </**div**>   <**table class="table table-bordered table-responsive"**>  <**thead**>  <**tr**>  <**th**>ID</**th**>  <**th**>部门名称</**th**>  <**th**>操作</**th**>  </**tr**>  </**thead**>  <**tbody**>  {% **for dep in deps** %}  <**tr**>  <**td**>{{ **dep**.**id** }}</**td**>  <**td**>{{ **dep**.**title** }}</**td**>  <**td**> *{# <a href="/dep/edit/?nid={{ dep.id }}" class="btn btn-success btn-xs">编辑</a>#}* <**a href="/dep/**{{ **dep**.**id** }}**/edit/" class="btn btn-success btn-xs"**>编辑</**a**>  <**a href="/dep/del/?nid=**{{ **dep**.**id** }}**" class="btn btn-danger btn-xs"**>删除</**a**>  </**td**>  </**tr**>  {% **endfor** %}  </**tbody**>  </**table**>  </**div**> </**div**> {% **endblock** %} |

## 4.然后我们把dep\_add.html改造为这个样子

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>新增部门</**title**> {% **endblock** %}  {% **block content** %}  <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>新建部门</**h3**>  </**div**>  <**div class="panel-body"**>  *{# 表单 #}* <**form action="/dep/add/" method="post"**>  {% **csrf\_token** %}  <**div class="form-group"**>  <**label**>部门名称</**label**>  <**div**>  <**input type="text" class="form-control" name="title" placeholder="输入部门"**>  </**div**>  </**div**>  <**button type="submit" class="btn btn-success"**>保 存</**button**>  </**form**>  </**div**>  </**div**>   </**div**> </**div**>  {% **endblock** %} |

## 5.然后我们把dep\_edit.html改造为下面的样子

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>编辑部门信息</**title**> {% **endblock** %}  {% **block content** %} <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>编辑部门</**h3**>  </**div**>  <**div class="panel-body"**>  *{# 表单 #}* <**form action="/dep/**{{ **dep**.**id** }}**/edit/" method="post"**>  {% **csrf\_token** %}  <**div class="form-group"**>  <**label**>部门名称</**label**>  <**div**> *{# <input type="hidden" class="form-control" name="id"value="{{ dep.id }}">#}* <**input type="text" class="form-control" name="title"value="**{{ **dep**.**title** }}**"**>  </**div**>  </**div**>  <**button type="submit" class="btn btn-success"**>更 新</**button**>  </**form**>  </**div**>  </**div**>   </**div**> </**div**> {% **endblock** %} |

## 6.然后我们在templates文件夹里面新建一个home.html页面,还是继承自layout.html

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>员工管理系统首页</**title**> {% **endblock** %}  {% **block content** %}  <**div class="container"**>  <**h2**>欢迎来到首页</**h2**>  <**ul**>  <**li class="btn btn-default"**><**a href="/dep/list/"**>部门列表</**a**></**li**>  </**ul**>  </**div**>  {% **endblock** %} |

## 7.然后我们修改一些views.py里面的home函数,让他渲染home.html页面给用户,这样子我们这个下面的界面就统一了

|  |
| --- |
|  |
|  |

## 8.我们可以修改一下index函数,让他也渲染home.html

|  |
| --- |
|  |

### 注意:如果你有些页面需要使用组件的样式而不是公共样式,或者需要引入指定是js这时候,我们就可以在模板母版里面添加员工{% **block css**%}{% **endblock** %}和员工{% **block js**%}{% **endblock** %}占位符,这样子,需要的页面就需要添加这两个block不需要的页面就不用添加非常灵活.可以提高开发效率.

# 2.我们继续来实现员工管理功能

## 1>在urls.py里面添加员工user/list/路由,对应views.py里面的user\_list函数

|  |
| --- |
|  |

## 2>然后我们需要在views.py里面添加员工user\_list函数,渲染员工user\_list.html模板

|  |
| --- |
|  |

## 3>为了让我们的模板能够显示数据,我们需要添加2条记录到用户数据表.

|  |
| --- |
|  |

## 4>在templates文件夹里面新建员工user\_list.html文件

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %} <**title**>用户列表</**title**> {% **endblock** %}  {% **block content** %} <**div class="container"**> <**div style="margin-bottom**: 5**px**;**"**>  <**a href="/user/add/" class="btn btn-primary"**>  <**span class="glyphicon glyphicon-plus-sign" aria-hidden="true"**></**span**>  新增用户  </**a**>  </**div**> <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**span class="glyphicon glyphicon-th-list" aria-hidden="true"**></**span**>  用户列表  </**div**>   <**table class="table table-bordered table-responsive"**>  <**thead**>  <**tr**>  <**th**>ID</**th**>  <**th**>姓名</**th**>  <**th**>密码</**th**>  <**th**>年龄</**th**>  <**th**>余额</**th**>  <**th**>入职时间</**th**>  <**th**>性别</**th**>  <**th**>所属部门</**th**>  </**tr**>  </**thead**>  <**tbody**>  {% **for user in users** %}  <**tr**>  <**td**>{{ **user**.**id** }}</**td**>  <**td**>{{ **user**.**name** }}</**td**>  <**td**>{{ **user**.**password** }}</**td**>  <**td**>{{ **user**.**age** }}</**td**>  <**td**>{{ **user**.**account** }}</**td**>  <**td**>{{ **user**.**create\_time**|**date**:**"Y-m-d H:i:s"** }}</**td**> *{# django中把日期转化为字符串的模板语法 #}* <**td**>{{ **user**.**get\_gender\_display** }}</**td**> *{# django提供的针对属性为数字特殊语法 get\_xxx\_display #}* <**td**>{{ **user**.**dep**.**title** }}</**td**> *{# django提供的针对外键为数字特殊语法 obj.xxx.xxx xxx就是xxx\_id去掉\_id #}* <**td**> *{# <a href="/dep/edit/?nid={{ dep.id }}" class="btn btn-success btn-xs">编辑</a>#}* <**a href="/user/**{{ **user**.**id** }}**/edit/" class="btn btn-success btn-xs"**>编辑</**a**>  <**a href="/user/del/?nid=**{{ **user**.**id** }}**" class="btn btn-danger btn-xs"**>删除</**a**>  </**td**>  </**tr**>  {% **endfor** %}  </**tbody**>  </**table**>  </**div**> </**div**> {% **endblock** %} |

## 5>然后我们在home.html文件里面给用户列表也添加员工链接

|  |
| --- |
|  |

### 运行程序,进入首页,效果如下

|  |
| --- |
|  |

## 6>点击员工列表,就会显示用户的列表

|  |
| --- |
|  |

## 7.我们还可以在layout.html里面添加员工跳转到用户管理页面的链接

|  |
| --- |
|  |

### 刷新页面,就会有这样的效果

|  |
| --- |
|  |

### 点击用户管理,就会跳转到用户列表页面

|  |
| --- |
|  |
|  |

## 8>我们注意到这个日期的格式不太好看,我们需要先把它出来一下,然后在渲染到页面.我们修改运行views.py里面的user\_list函数

|  |
| --- |
|  |

### 此时运行程序,页面中的日期就好看多了

|  |
| --- |
|  |

### 然后我们还有一个问题,就是性别应该显示男或者女,不要显示0和1因为用户不知道0和1代表什么意思.

## 9.django已经考虑到这个问题了,它也专门为性别封装了应该函数叫做get\_geder\_display方法,我们可以用这个方法来代替gender属性.其实所有使用代码来表示的属性都有这个get\_xxx\_display方法来获取你设置的值.然后我们也可以在模板在使用不过不需要小括号

|  |
| --- |
|  |

### 效果如下

|  |
| --- |
|  |

### 然后我们还有一个问题,就是所属部门是数字,也是不好的.我们想它显示部门名称,这个怎么弄?

## 10.django也考虑到了这一点,这就是为什么它的数据库里面会在我们的外键上面添加应该\_id,此时我们只需要把这个\_id去掉,就可以获取到它这个id对应的一行数据封装为应该对象返回给你,然后你就可以使用.xx来获取你需要的数据这里我们dep\_id是外键,我们就可以使用user.dep.title来获取title的值,我们把user\_list.html模板文件修改一下.

|  |
| --- |
|  |

### 刷新页面,此时看上去就舒服多了

|  |
| --- |
|  |

## 11.django模板语法也有提供把日期转化为字符串的模板语法,我们可以把views.py里面的user\_list函数中手动转换的代码注释了,使用模板语法

|  |
| --- |
|  |
|  |

### 运行程序,效果是一样的

|  |
| --- |
|  |

## 12.然后我们来学习条件用户的功能,注意django提供了Form组件和ModelForm组件,用ModelForm组件是最方便的.当然我们需要了解用最原始的方法怎么操作,因为我们在user\_list.html文件里面已经指定了新增用户是路由的/user/add/,我们需要先在urls.py里面添加一个user/add/路由,和views.py里面的user\_add函数对应起来

|  |
| --- |
|  |

## 13.然后我们需要在views.py里面添加一个user\_add函数,先出来get请求,渲染一个叫做user\_add.html的文件然后返回给用户

|  |
| --- |
|  |

## 14.然后我们需要在templates文件夹里面创建这个user\_add.html文件,注意:性别和部门需要使用下拉框让用户选择,因为我们这里规定了男是1女是0,部门必须是我们数据库部门表里面有的部门.不能让用户随便写.最后从数据库里面拿到的数据.我们修改运行user\_add函数,把性别和部门信息传递给模板文件

|  |
| --- |
|  |

## 15.然后我们来创建一个user\_add.html文件,内容如下

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>新增用户</**title**> {% **endblock** %}  {% **block content** %}  <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>新建用户</**h3**>  </**div**>  <**div class="panel-body"**>  *{# 表单 #}* <**form action="/user/add/" method="post"**>  {% **csrf\_token** %}  <**div class="form-group"**>  <**label**>姓名</**label**>  <**div**>  <**input type="text" class="form-control" name="name" placeholder="输入姓名"**>  </**div**>  </**div**>  <**div class="form-group"**>  <**label**>密码</**label**>  <**div**>  <**input type="text" class="form-control" name="pwd" placeholder="密码"**>  </**div**>  </**div**>  <**div class="form-group"**>  <**label**>年龄</**label**>  <**div**>  <**input type="text" class="form-control" name="age" placeholder="年龄"**>  </**div**>  </**div**>  <**div class="form-group"**>  <**label**>余额</**label**>  <**div**>  <**input type="text" class="form-control" name="acc" placeholder="余额"**>  </**div**>  </**div**>  <**div class="form-group"**>  <**label**>入职时间</**label**>  <**div**>  <**input type="text" class="form-control" name="ctime" placeholder="入职时间"**>  </**div**>  </**div**>  <**div class="form-group"**>  <**label**>性别</**label**>  <**div**>  <**select name="gender" class="form-control"**>  {% **for ch in g\_choice** %}  <**option value="**{{ **ch**.0 }}**"**>{{ **ch**.1 }}</**option**>  {% **endfor** %}  </**select**>  </**div**>  </**div**>  <**div class="form-group"**>  <**label**>部门</**label**>  <**div**>  <**select name="dep" class="form-control"**>  {% **for dep in deps** %}  <**option value="**{{ **dep**.**id** }}**"**>{{ **dep**.**title** }}</**option**>  {% **endfor** %}  </**select**>  </**div**>  </**div**>  <**button type="submit" class="btn btn-success"**>保 存</**button**>  </**form**>  </**div**>  </**div**>   </**div**> </**div**>  {% **endblock** %} |

### 界面长这样子

|  |
| --- |
|  |

## 16.然后我们在user\_add函数里面添加处理post请求的代码

|  |
| --- |
|  |

### 测试一下

|  |
| --- |
|  |
|  |

### 添加成功,不过这种方式非常不好,有4大缺点

|  |
| --- |
|  |

## 现在我们需要学习使用django提供的表单组件来做,会比较简单,下面就是他的基本语法

|  |
| --- |
|  |

## 可以看到form组件还是比较麻烦,我们再学习另外一个叫做ModelForm的组件,假如我们有一个这样子的model

|  |
| --- |
|  |

## 那么ModelForm就可以直接使用model里面的字段

|  |
| --- |
|  |

## ModelForm还支持自己定义字段

|  |
| --- |
|  |

## 17.下面我们先来改造user\_add函数,需要先导入ModelForm和定义一个类

|  |
| --- |
|  |

### 然后,我们写一个类继承自forms.ModelFom

|  |
| --- |
|  |

### 注意:其实不一定要把数据表的所有列都用上,有时候我们可能只需要使用某些列,这时候就可以根据需要在fields里面填写需要的列

### 然后需要把form作为参数传递给模板

|  |
| --- |
|  |

## 18.然后我们在模板文件里面变量这个form对象,就可以完成表单的填写,这里我们暂时没有使用模板继承

|  |
| --- |
|  |

### 运行程序效果如下

|  |
| --- |
|  |

### 可见还存在一个问题,就是部门信息显示为一个对象,好像不太好.如何解决?

## 19因为获取到的是Department类的对象,我们需要在这个Department类中定义一个特殊方法\_\_str\_\_(self),在里面返回title属性

|  |
| --- |
|  |

### 然后刷新页面,发现显示正常了

|  |
| --- |
|  |

### 那么我们怎么让这个页面也有其他页面的样式呢?

## 20.我们还是让他继承layout.html.先把user\_add.html代码写成下面的样子

|  |
| --- |
| {% **extends 'layout.html'** %}  {% **block title** %}  <**title**>新增用户</**title**> {% **endblock** %} {% **block content** %}  <**div**>  <**div class="container"**>  <**div class="panel panel-default"**>  *<!-- Default panel contents -->* <**div class="panel-heading"**>  <**h3 class="panel-title"**>新建用户</**h3**>  </**div**>  <**div class="panel-body"**>  *{# 表单 #}* <**form action="/user/add/" method="post"**>  {% **csrf\_token** %}  {% **for item in form** %}  <**div class="form-group"**>  {{ **item**.**label** }}  {{ **item** }}  </**div**>  {% **endfor** %}   <**button type="submit" class="btn btn-success"**>保 存</**button**>  </**form**>  </**div**>  </**div**>   </**div**> </**div**>   {% **endblock** %} |

### 运行的效果如下

|  |
| --- |
|  |

## 21.还是很丑,此时我们需要在ModelForm的类里面设置widgets属性

|  |
| --- |
|  |

### 然后刷新页面,效果出来了

|  |
| --- |
|  |

### 但是这样子非常麻烦,我们有更好的方法,就是定义一个函数来完成

|  |
| --- |
|  |

### 然后效果就实现了

|  |
| --- |
|  |

### 还可以添加占位符placeholder,注意字段不要有默认值如果有设置,把它删除.

|  |
| --- |
|  |

### 效果如下:

|  |
| --- |
|  |

### 不过对于下拉列表我们是可以给他指定默认值的.他会自动选择第一项作为默认值

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 那么问题来了,我们如何在views.user\_add函数里面获取这个表单的数据呢?

## 22.此时我们如果要在views.user\_add函数里面获取这个表单的数据,我们可以使用以前的方法,但是名称必须严格按照你定义的类的字段名称来获取,否则会报错

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |
|  |

## 23.其实我们在提交数据之前,需要利用ModelForm的校验功能来进行数据校验,然后如果校验成功,用form的保存方法就可以很轻松的保存数据.

|  |
| --- |
|  |

### 测试一下

|  |
| --- |
|  |
|  |

## 24.这个只是校验成功的情况,万一校验失败其实不应该让他跳转,需要重新渲染user\_add.html页面,并且传递校验后的form.在页面上显示错误信息所以我们修改下面代码

|  |
| --- |
|  |

## 25.然后我们还需要在user\_add.html页面里面显示错误,注意需要改变form的默认校验信息

|  |
| --- |
|  |

### 我们进入user/add/页面,不填写然任何数据,任何点击保存,效果如下

|  |
| --- |
|  |

## 26上面的验证只能够校验是否为空,如果你还需要其他校验,你需要重新定义这个字段

|  |
| --- |
|  |

### 然后效果是这样子的

|  |
| --- |
|  |

### 也就是说,某些字段有多个验证的,需要另外重写,其他的使用默认就好了.

## 27.另外,错误信息不仅仅可以用英文显示,也可以用中文显示.我们来设置一下settings.py

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |

### 还有一点瑕疵,就是入职时间哪里可以用一个日期选择器,我们以后会完成这个功能

# 扩展: datetime类型的数据转化为字符串

|  |
| --- |
| 8 |