KylinBlog开发注解-Part6

表单设计与使用

- 表单用于让用户提交数据或上传文件，也用于用户编辑已有数据。
- Django中的表单Form类的作用是把用户输入的数据转化成Python对象格式，以便后续操作

自定义表单类

Django提供了两种自定义表单的方式：继承 `Form类` 和 `ModelForm类`

- 继承 `Form类` 需要自定义表单中的字段
- 继承 `ModelForm类` 可以根据Django模型自动生成表单

```
# app/forms.py
# 自定义表单字段
from django import forms
from .models import Contact

class ContactForm1(forms.Form):
    name = forms.CharField(label="Your Name", max_length=255)
    email = forms.EmailField(label="Email address")

# 根据模型创建
class ContactForm2(forms.ModelForm):

    class Meta:
        model = Contact
        fields = ('name', 'email',)
```

- Django模型models中用 `verbose_name` 来给字段添加一个别名或描述，而表单使用 `label`
- 自定义表单常置于app目录下的 `forms.py` 中，这样方便集中管理表单
- 自定义表单类似于库文件，直接import导入即可

自定义字段错误信息

对于每个字段除了可以设置 是否必需、最大长度、最小长度外，还可以针对每个字段自定义验证错误信息，如下：

```
from django import forms

class LoginForm(forms.Form):
    username = forms.CharField(
        required=True,
        max_length=20,
        min_length=6,
        error_messages={
            'required': '用户名不能为空',
            'max_length': '用户名长度不得超过20个字符',
            'min_length': '用户名长度不得少于6个字符',
        }
    )
    password = forms.CharField(
        required=True,
        max_length=20,
```

```
        min_length=6,
        error_messages={
            'required': '密码不能为空',
            'max_length': '密码长度不得超过20个字符',
            'min_length': '密码长度不得少于6个字符',
        }
    )
```

表单实例化与初始化

- 定义好一个表单类之后，就可以对其进行实例化或初始化
- `form=ContactForm() #空表单`

- 有时需要对表单设置一些初始数据，可以通过`initial`方法或者设置`default_data`来实现
- ```
  # initial方法初始化
  form = ContactForm(
      initial={
          'name': 'First and Last Name',
      },)

  # default_data默认值
  default_data = {'name': 'John', 'email': 'someone@hotmail.com', }
  form = ContactForm(default_data)
  ```

- 用户提交的数据可以通过以下方法与表单结合，生成与数据结合过的表单。
- `form = ContactForm(data=request.POST, files=request.FILES)`

KylinBlog中的表单实例

注册功能的实现,没有使用forms功能，使用HTML5实现

register.html

```
{%load static%}
<!DOCTYPE html>
<html lang="zh-hans">
<head>
    <meta charset="UTF-8">
    <title>注册</title>
    <link rel="shortcut icon" href="{% static 'css/images/gt_favicon.png' %}">
    <link rel="stylesheet" href="{%static 'css/log.css' %}">
    <link rel="stylesheet" href="{%static 'css/semantic.css' %}">
</head>
<body class="register">

            <h1 class="ui teal header"><font color="black">KylinBlog-用户注册</font></h1>

                    <form class="ui form" method="post" action="{%url 'register'%}"
 enctype="multipart/form-data">

                            <input type="text" name="username" placeholder="请输入用户名"><br>

```
```

```
                        <div class="field">
                            <input type="password" name="password_1" placeholder="请输入密码"><br>
                        </div>
                        <div class="field">
                            <input type="password" name="password_2" placeholder="请确认密码"><br>
                        </div>
                        {{ error }}<br>
                        <button class="ui fluid large teal button" type="submit">注册</button>
                    </form>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

- `<input type="text" name="username" placeholder="请输入用户名"><br>`
  - 输入控件，并将输入的变量命名为 `username`
- `<input type="password" name="password_1" placeholder="请输入密码"><br>`
  - 输入控件，并将输入的变量命名为 `password_1`
- `<button class="ui fluid large teal button" type="submit">注册</button>`
  - 按钮控件 `submit` 方法，上传的请求为 `POST请求`

views.py

```
# 手写注册视图view模型
def register(request):
    if request.method =='POST':
        user_name = request.POST.get('username', '')
        pass_word_1 = request.POST.get('password_1', '')
        pass_word_2 = request.POST.get('password_2', '')

        if User.objects.filter(username=user_name):
            return render(request, 'register.html', {'error': '用户已存在'})
        if pass_word_1 != pass_word_2:
            return render(request, 'register.html', {'errpr': '两次密码不一致'})
        user = User()
        user.username=user_name
        user.password=make_password(pass_word_1)
        user.is_staff=1
        user.is_superuser=0
        user.is_active=1
        user.save()
        return render(request,'index.html')
    return render(request,'register.html')
```

- 基于函数的视图view实现方法
- `if request.method =='POST':` 过滤条件，只筛选方法为 `POST` 所上交的数据
- ```
  user_name = request.POST.get('username', '')
  pass_word_1 = request.POST.get('password_1', '')
  pass_word_2 = request.POST.get('password_2', '')
  ```
  - 通过 `get` 方法来获得表单上交的数据
  ```

- ```
  if User.objects.filter(username=user_name):
              return render(request, 'register.html', {'error': '用户已存在'})
          if pass_word_1 != pass_word_2:
              return render(request, 'register.html', {'errpr': '两次密码不一致'})
  ```
  - 在数据库中筛选，看是否有重复现象
- ```
  user = User()
  user.username=user_name
  user.password=make_password(pass_word_1)
  user.is_staff=1
  user.is_superuser=0
  user.is_active=1
  user.save()  # 使用save()方法保存到数据库中
  ```