Django Form表单Field属性与方法

前面通过《Django表单系统初体验》一节，我们初步认识了 Django 的表单系统，然后通过《Django HTML表单实例应用》一节对于传统的 HTML 表单也有了更深入的认识。

如果是处理简单的表单，那么可以使用 HTML 表单的形式，但是在实际的业务处理中，表单往往都比较复杂，可能同时对多个字段进行校验，如果再按照 HTML 表单的形式为每个字段编写模板代码，并在视图函数中完成校验给出错误提示，就会非常麻烦。所以 Django为解决这个问题提供了表单系，它让开发者更注重于实现业务逻辑，而不是编写反复重复的代码。本节我们将从表单系统的字段属性讲起，让我们一起走进它的世界吧。

1. forms定义表单

1) 使用表单系统定义表单

表单系统的核心是 Form 对象，它将表单中字段封装成一系列的 Field 字段和验证规则，以此来自动生成 HTML 表单标签。和其他对象的定义规则类似，我们可以将 Form 对象定义在任何位置，不过 Django 建议最好 Form 对象定义在应用的 forms.py 中，就像视图函数写在 views.py 文件中一样，其实这样做的主要原因是为了实现分类管理。所以我们要在 index 应用下新建一个 forms.py 文件。

```
1. from django import forms
2. class TitleSearch(forms.Form):
3.     title=forms.CharField(label='书名')
```

上述代码中，首先 Django 规定所有的 Form 对象都必须继承自 django.forms.Form，然后定义了一个 title 属性，它是 forms.CharField 类型的 Field，根据名字可以得知，它将 title 指定为字符类型， 而 lable 标签指定了这个字段的名称，在此处我们省略一个它的默认属性即 required，默认值为 True，代表是必填项。

从上述的介绍中，我们可以看出它和我们之前学习的 Model 模型类非常的相似，这我们在《Django表单系统初体验》一节也提到过，大家可以进行比较学习。

2. 表单字段基类Field属性

Django 的表单系统同 Model 模型类一样，内置了数十种的表单字段类型，当这些内置的字段无法满足场景需求的时，Django 同样支持自定义 Field 类。Field 类定义在 django/forms/fields.py 文件中，在 Field 的构造函数中定义很多属性，Field 的子类(即字段)中也可以根据需要设定这些属性值，它的常用属性如下所示：

1) required

设定当前的 Field 是否是必须提供的，默认值是 True，即必须提供。Field 中定义了一个类属性 empt y_values，clean 方法会判断当前 Field 的值是否是空值，若当值是 None 或者空字符串等空值时就会抛出 ValidationError 异常。如果将其设置为 False，那么 Field 将会变成可选的，即使不提供也不会抛出异常。

2) widget

指定在页面中字段的控件（插件）类型，可以是 Widget 类或者 Widget 类实例。默认的控件使用 TextInput（相当于 HTML 表单中 <input type="text">）。实例如下所示：

```
1. from django import forms
2. class TitleSearch(forms.Form):
3.     title=forms.CharField(label='书名',widget=forms.Textarea)#文本域
```

3) lable

指定在页面中显示的字段的名称提示。在定义TitleSearch 时已经使用了，将 title 的 label 属性设定为“”书名“”。如果不指定的话，页面中将直接显示字段定义的变量名。

4) initial

指定字段初始值，默认值为 None。当给字段的 initial 属性设定一个非空值时，页面中的对应表单将使用这个值填充。举例如下给 title 设定初始值

```
1. from django import forms
2. class TitleSearch(forms.Form):
3.     title=forms.CharField(label='书名',initial="C语言中文网")
```

5）error_messages

这个属性用于覆盖 Field 默认的错误消息。为了更好地说明它的作用，下面用 title 字段举例说明，如下所示：

```
#不指定该属性的情况下
In [2]: from django import forms
In [3]: title=forms.CharField(lable="title name")
In [4]: title.clean('')
.....
ValidationError: ['这个字段是必填项。']
#指定该属性
In [7]: title=forms.CharField(label="title name",error_messages={"required":"请输入正确的title"})
In [8]: title.clean('')
....
ValidationError: ['请输入正确的title']
```

6) disabled

其默认值是 False，如果修改为 True，则当前的表单字段将不可编辑。当设置字段为不可编辑时，需要提供初始值（initial），否则，这个字段也就没有意义了。

3. 表单字段基类Field方法

1) clean()

上面介绍了一些常用属性，在这里重点介绍 Field 常用的一些方法。Field 实例调用 clean 方法用来对传递的数据做“清理”和校验：对数据做清理可以将数据转换成对应的 Python 对象；而校验则是检验当前给定的数据是否满足 Field 属性的约束，如果不满足，则会抛出 ValidationError 异常（验证失败），如下所示：

```
#输入有效的Email通过clean验证
In [2]: from django import forms
   ...: f = forms.EmailField()
   ...: f.clean('Mr@163.com')
Out[2]: 'Mr@163.com'
#输入无效的Email通过clean验证并抛出异常
In [3]: f.clean('无效的地址')
....
ValidationError: ['输入一个有效的 Email 地址。']
```

每个 Field 的实例都有一个 clean() 方法，它接受一个参数，然后返回“清洁的”数据或者抛出一个 django.forms.ValidationError 异常。在后续会对 Form 表单的校验做针对的讲解。

提示：clean()在 Django 的表单系统中非常重要，该方法经常用在开发或测试过程中对数据进行验证和测试。
在下一节，我们将介绍 Django 表单系统的常用字段。