1. MultiValueField

在一些个性化的需求下，用户的一个属性可能是多条数据的组合。比如用户的电话号码 639980-004，区分为主机号码 639980 和分机号码 004 两条数据，但是它们都属于同一个用户属性，所以需要通过定义属性字段将两条数据整合起来进行开发，此时就要用到多值操作属性字段 MultiValueField。自定义属性字段如下：

```
class PhoneField(forms.MultiValueField):
    '''自定义电话号码属性字段，可以同时接收两条数据'''
    def __init__(self, **kwargs):
        error_messages = {
            'error_msg': '电话号码输入有误，请检查后重新输入'
        }
        fields = {
            forms.CharField(
                error_messages={'error_msg': '主机号码输入有误'},
                validators=[
                    RegexValidator(r'^\d{8}$', '请输入有效的主机号码')
                ]
            ),
            forms.CharField(
                error_messages={'error_msg': '分机号码输入有误'},
                validators=[
                    RegexValidator(r'^\d{3}$', '请输入有效的分机号码')
                ]
            )
        }
        super().__init__(error_messages=error_messages,
                         fields=fields,
                         require_all_fields=False,
                         **kwargs)
```

完成自定义多值操作属性字段后，就可以在表单实例中直接声明属性了，但是一定要重新添加 widget 限定规则，修改为 django.forms.MultiWidget 组件才能使用。不过，在实际项目中，一般通过数据库建模来解决属性聚合的问题，通过代码直接封装该属性字段的方式并不是很友好。

2. ModelChoiceField

这是一个非常有用的属性字段，在项目中经常会遇到在页面打开的同时就需要初始化并展示数据，通过下拉列表框进行选择等情况，如选择文章的发表类型（原创、翻译、转载），就需要在页面打开的同时在表单中渲染展示数据，Django 提供的 ModelChoiceField 属性字段就是专门用于处理这种情况的，代码如下：

```
class MyForm(forms.Form):
    ......
    source = forms.ModelChoiceField(queryset=ArticleSource.objects.all(),
                                    empty_label='--choice--')
```

Django 框架直接渲染该属性字段得到如下网页标签：