Django表单系统工作原理详述

通过继承 Form 对象，定义所需要的表单字段，基本上完成了表单的定义。它可以自动生成 HTML，完成字段值的校验，并给出相应错误的提示信息。本节介绍这些功能的实现过程中的工作原理。

1. 表单对象的创建过程

我们知道所有的表单对象都继承自 Form，首先来看 Form 的定义如下所示：

```
1. class Form(BaseForm, metaclass=DeclarativeFieldsMetaclass):
```

Form 中指定了基类 BaseForm 和元类 DeclarativeFieldsMetaclass。BaseForm 中定义了生成 HTML 与字段值校验的方法，而 DeclarativeFieldsMetaclass 则定义了创建 Form 对象的过程。DeclarativeFieldsMetaclass 在 Django 中的源码实现如下所示：

```
1. class DeclarativeFieldsMetaclass(MediaDefiningClass):
2.     def __new__(mcs, name, bases, attrs):
3.         # 收集基类中声明的字段。
4.         current_fields = []
5.         #遍历当前类中定义的属性
6.         for key, value in list(attrs.items()):
7.             #只添加field类型的实例
8.             if isinstance(value, Field):
9.                 current_fields.append((key, value))
10.                attrs.pop(key)
11.        attrs['declared_fields'] = OrderedDict(current_fields)
12.         #调用父类的new方法创建类对象
13.        new_class = super(DeclarativeFieldsMetaclass, mcs).__new__(mcs, name, bases, attrs)
14.
15.        # Walk through the MRO.
16.        declared_fields = OrderedDict()
17.        for base in reversed(new_class.__mro__):
18.            # 继承父类的字段定义
19.            if hasattr(base, 'declared_fields'):
20.                declared_fields.update(base.declared_fields)
21.
22.            for attr, value in base.__dict__.items():
23.                if value is None and attr in declared_fields:
24.                    declared_fields.pop(attr)
```

```
        #在这里创建的类对象添加了base_fields和declared_fields两个属性
        new_class.base_fields = declared_fields
        new_class.declared_fields = declared_fields
        return new_class
```

DeclarativeFieldsMetaclass 继承自 MediaDefiningClass，并调用了它的 new 方法创建了类对象。MediaDefiningClass 的实现如下：

```
class MediaDefiningClass(type):
    "元类用于具有媒体定义的类。"
    def __new__(mcs, name, bases, attrs):
        new_class = super(MediaDefiningClass, mcs).__new__(mcs, name, bases, attrs)
        #如果属性中没有media, 则通过media_property
        if 'media' not in attrs:
            new_class.media = media_property(new_class)
        return new_class
```

MediaDefiningClass 中并没有做太多的工作，只是给类对象添加了 media 属性，当我们使用 JavaScript 和 CSS 的时候。这个 media 属性起到引用的作用。最后在类对象返回之前，给它附加了两个属性，且都指向了 declared_fields，它是一个 OrderedDict 类型的实例，存储了表单中定义的 Field。例如，可以查看 TitleSearch 的 base_fields 属性：

```
In [1]: from index.forms import TitleSearch
In [2]: TitleSearch.base_fields
Out[2]: OrderedDict([('title', <django.forms.fields.CharField at 0x87f9a30>)])
```

2. 表单对象生成HTML的实现过程

我们知道，在 CMD 命令行工具中打印 Form 对象实例可以看到自动生成的 HTML，所以，可以将 Form 对象传递到模板中替换模板变量。在 Python 中使用 print 方法打印对象实例，会调用 __str__ 方法，所以，表单对象生成 HTML 的过程就在这个方法中实现。__str__ 方法定义如下：

```
def __str__(self):
    return self.as_table()
```

其中只是调用了 as_table 方法，它的实现如下：

```
def as_table(self):
    "得到一张HTML table格式"
    return self._html_output(
        normal_row='<tr%(html_class_attr)s><th>%(label)s</th><td>%(errors)s%(field)s%(help_text)s</td></tr>',
        error_row='<tr><td colspan="2">%s</td></tr>',
        row_ender='</td></tr>',
```

```
            help_text_html='<br><span class="helptext">%s</span>',
            errors_on_separate_row=False,
        )
```

代码中的 _html_output 方法实现表单对象到 HTML 的转换，除了 as_table 外，as_url 和 as_p 的实现同样也使用了该方法。

_html_output 方法接受 5 个参数，用来定义 HTML 的格式，如 normal_row 用来定义字段的样式与信息、row_ender 定义了结束标签以及 help_text_html 定义了字段帮助信息等。该方法的实现源代码较多，大家可以自己查看源码，总体来说它的主要作用就是解析表单对象中定义的各个 Field，给每个 Field 生成表单 HTML，最后对每个字段的 HTML 拼接得到完整的表单。

对于 Django 表单系统工作原理的了解，更有助于我们掌握 Django 表单系统的应用，我们学会用看源码的方式去加强对框架的理解，这是一种不错的选择，希望各位小伙伴都能够有所收获。