# Django Form表单自定义验证规则

在《Django Form表单实现自定义字段》我们讲解了如何是实现 Django Form 表单自定义表单字段，在大多数情况自定义 Form 表单字段的同时都会添加额外的数据校验逻辑，但是如果只是为了添加校验逻辑，而单独再去定义一个字段，也就有点事半功倍了，所以，如果只需要对一些表单字段做额外的检验，可以将检验逻辑编写在定义的 Form 类中，以类方法形式存在。

## 1. 实现自定义校验规则

表单系统会自动查找以 clean_ 开头，以字段名结尾的方法，它会在验证字段合法性的过程中被调用，因此，如果想要自定义校验逻辑，可以按如下方式编写代码：

```
class RegForm(forms.Form):
    name = forms.CharField(label='用户名')

    def clean_name(self):
        name=self.cleaned_data['name']
        if len(name)<6:
            raise forms.ValidationError("你注册的用户名字符太短了")
        return name
```

上述代码中，我们依然首先定义了一个 表单 Form 类，然后并定义了一个字段 name，并在该类中额外添加了 name 字段的验证逻辑即通过 clean_name 方法实现，该方法会在 name 字段的默认验证逻辑执行完成后执行，所以，可以直接通过 cleaned_data 属性获取到符合校验要求的数据值。

在 clean_name 方法中只是简单的校验当前的数据长度不能小于 6 ，若不符合要求则会抛出异常 ，并给出错误提示信息，需要注意的是在自定义验证方法结束时，需要将字段的值返回，否则这个字段的值就会变成 None，这个地方需要大家在自定义校验规则的时候特别注意，这也是常见的错误。

下面我们就来看看验证逻辑是否能够生效吧，打开 Django 的 shell 环境，进行如下操作：

```
In [1]: from index.forms import RegForm
In [2]: form=RegForm
In [3]: form=RegForm({'name':'cyuyan'})
```

```
In [4]: form.is_valid()
Out[4]: True
In [5]: form=RegForm({'name':'cyuyanzhongwenwang'})
In [6]: form.is_valid()
Out[6]: True
In [7]: form=RegForm({'name':'cyu'})
In [8]: form.is_valid()
Out[8]: False
In [9]: form["name"].errors
Out[9]: ['你注册的用户名字符太短了']
```

从上面的测试可以看出，通过自定义 clean_name 方法实现了自定义的校验逻辑。

本节的内容不多，也比较简单，Django 提供的自定义验证规则，能够使开发者更加方便自如的实现业务需求，也希望各位小伙伴在使用 Django Form 表单系统的时候，要敢于尝试，程序开发虽然要遵循一定的规则，但是你的思路要敢于天马行空，这样才能不断提升自己能力，不断在探索和尝试中进步，学习到新的知识。