为了防止机器人频繁登录网站或者破坏分子恶意登录，很多用户登录和注册系统都提供了图形验证码功能。

验证码（CAPTCHA）是“Completely Automated Public Turing test to tell Computers and Humans Apart”（全自动区分计算机和人类的图灵测试）的缩写，是一种区分用户是计算机还是人的公共全自动程序。可以防止恶意破解密码、刷票、论坛灌水，有效防止某个黑客对某一个特定注册用户用特定程序暴力破解方式进行不断的登陆尝试。

图形验证码的历史比较悠久，到现在已经有点英雄末路的味道了。因为机器学习、图像识别的存在，机器人已经可以比较正确的识别图像内的字符了。但不管怎么说，作为一种防御手段，至少还是可以抵挡一些低级入门的攻击手段，抬高了攻击者的门槛。

在Django中实现图片验证码功能非常简单，有现成的第三方库可以使用，我们不必自己开发（也要能开发得出来，囧）。这个库叫做django-simple-captcha。

**一、安装captcha**

在Pycharm的terminal中，首先进入mysite\_env虚拟环境，然后安装第三方库：

执行命令：pip install django-simple-captcha

(mysite\_env) F:\Django\_course\mysite\_env\Scripts>pip install django-simple-captcha

Collecting django-simple-captcha

Downloading django-simple-captcha-0.5.5.zip (144kB)

100% |████████████████████████████████| 153kB 139kB/s

Requirement already satisfied: setuptools in f:\django\_course\mysite\_env\lib\site-packages (from django-simple-captcha)

Collecting six>=1.2.0 (from django-simple-captcha)

Downloading six-1.11.0-py2.py3-none-any.whl

Requirement already satisfied: Django>=1.7 in f:\django\_course\mysite\_env\lib\site-packages (from django-simple-captcha)

Collecting Pillow>=2.2.2 (from django-simple-captcha)

Downloading Pillow-4.3.0-cp36-cp36m-win\_amd64.whl (1.5MB)

100% |████████████████████████████████| 1.5MB 364kB/s

Requirement already satisfied: pytz in f:\django\_course\mysite\_env\lib\site-packages (from Django>=1.7->django-simple-captcha)

Collecting olefile (from Pillow>=2.2.2->django-simple-captcha)

Building wheels for collected packages: django-simple-captcha

Running setup.py bdist\_wheel for django-simple-captcha ... done

Stored in directory: C:\Users\Administrator\AppData\Local\pip\Cache\wheels\f2\a1\ce\6eddab991940a2502d2252870ce5870388ac085e9508bcc2cb

Successfully built django-simple-captcha

Installing collected packages: six, olefile, Pillow, django-simple-captcha

Successfully installed Pillow-4.3.0 django-simple-captcha-0.5.5 olefile-0.44 six-1.11.0

Django自动帮我们安装了相关的依赖库six、olefile和Pillow，其中的Pillow是大名鼎鼎的绘图模块。

**二、注册captcha**

在settings中，将‘captcha’注册到app列表里：

# Application definition

INSTALLED\_APPS = [

'django.contrib.admin',

'django.contrib.auth',

'django.contrib.contenttypes',

'django.contrib.sessions',

'django.contrib.messages',

'django.contrib.staticfiles',

'login',

'captcha',

]

captcha需要在数据库中建立自己的数据表，所以需要执行migrate命令生成数据表：

(mysite\_env) F:\Django\_course\mysite>python manage.py migrate

Operations to perform:

Apply all migrations: admin, auth, captcha, contenttypes, login, sessions

Running migrations:

Applying captcha.0001\_initial... OK

**三、添加url路由**

在根目录下的urls.py文件中增加captcha对应的地址：

from django.contrib import admin  
from django.urls import path, include  
from login import views  
  
  
  
urlpatterns = [  
 path(**'admin/'**, admin.site.urls),  
 path(**'index/'**, views.index),  
 path(**'login/'**, views.login),  
 path(**'register/'**, views.register),  
 path(**'logout/'**, views.logout),  
 path(**'captcha/'**, include(**'captcha.urls'**))  
]

由于使用了二级路由机制，需要在顶部from django.urls import path加上 include。

## 四、修改forms.py

如果上面都OK了，就可以直接在我们的forms.py文件中添加CaptchaField了。

**from** django **import** forms

**from** captcha.fields **import** CaptchaField

**class** **UserForm**(forms**.**Form):

username **=** forms**.**CharField(label**=**"用户名", max\_length**=**128, widget**=**forms**.**TextInput(attrs**=**{'class': 'form-control'}))

password **=** forms**.**CharField(label**=**"密码", max\_length**=**256, widget**=**forms**.**PasswordInput(attrs**=**{'class': 'form-control'}))

captcha **=** CaptchaField(label**=**'验证码')

注意需要提前导入from captcha.fields import CaptchaField，然后就像写普通的form字段一样添加一个captcha字段就可以了！

## 五、修改login.html

由于我们前面是手动生成的form表单，所以还要修改一下，添加captcha的相关内容，如下所示：

{% extends 'base.html' %}

{% load staticfiles %}

{% block title %}登录{% endblock %}

{% block css %}<link href**=**"{% static 'css/login.css' %}" rel**=**"stylesheet"/>{% endblock %}

{% block content %}

<div class**=**"container">

<div class**=**"col-md-4 col-md-offset-4">

<form class**=**'form-login' action**=**"/login/" method**=**"post">

{% if message %}

<div class**=**"alert alert-warning">{{ message }}</div>

{% endif %}

{% csrf\_token %}

<h2 class**=**"text-center">欢迎登录</h2>

<div class**=**"form-group">

{{ login\_form.username.label\_tag }}

{{ login\_form.username}}

</div>

<div class**=**"form-group">

{{ login\_form.password.label\_tag }}

{{ login\_form.password }}

</div>

<div class**=**"form-group">

{{ login\_form.captcha.errors }}

{{ login\_form.captcha.label\_tag }}

{{ login\_form.captcha }}

</div>

<button type**=**"reset" class**=**"btn btn-default pull-left">重置</button>

<button type**=**"submit" class**=**"btn btn-primary pull-right">提交</button>

</form>

</div>

</div> *<!-- /container -->*

{% endblock %}

这里额外增加了一条{{ login\_form.captcha.errors }}用于明确指示用户，你的验证码不正确。

## 六、查看效果

重启服务器，进入登录页面，尝试用用户名错误、密码不对、验证码不对、全对的不同情况，看看我们新增的四位验证码的效果如何。



就是这么简单！我们加入了一个防止机器人或者恶意登录的图形验证码功能，虽然界面难看了点，但底子是好的，你可以根据需要进行美化。其中验证图形码是否正确的工作都是在后台自动完成的，只需要使用is\_valid()这个forms内置的验证方法就一起进行了，完全不需要在视图函数中添加任何的验证代码，非常方便快捷！

关于captcha的功能，当然绝不仅限于此，你可以设置六位、八位验证码，可以对图形噪点的生成模式进行定制，这些就留待你自己学习和研究了。