Django Simple Captcha Documentation

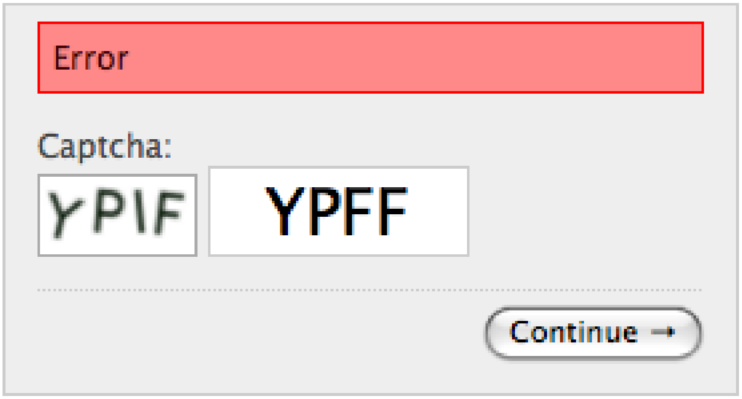
Release 0.4.6

Marco Bonetti

September 15, 2015

Contents

Django Simple Captcha is an extremely simple, yet highly customizable Django application to add captcha images to any Django form.



CHAPTER 1

Features

- Very simple to setup and deploy, yet very configurable
- Can use custom challenges (e.g. random chars, simple maths, dictionary word, ...)
- Custom generators, noise and filter functions alter the look of the generated image
- Supports text-to-speech audio output of the challenge text, for improved accessibility
- Ajax refresh

CHAPTER 2

Requirements

- Django 1.4+
- A recent version of the Python Imaging Library (PIL 1.1.7 or Pillow 2.2+) compiled with FreeType support
- Flite is required for text-to-speech (audio) output, but not mandatory

CHAPTER 3

Python 3 compatibility

The current development version supports Python3 via the six compatibility layer. You will need to install Pillow HEAD because PIL doesn't support Python3 yet.

CHAPTER 4

Contents:

4.1 Using django-simple-captcha

4.1.1 Installation

1. Download `django-simple-captcha` using pip by running: `pip install django-simple-captcha`
2. Add `captcha` to the `INSTALLED_APPS` in your `settings.py`
3. Run `python manage.py syncdb` (or `python manage.py migrate` if you are managing database migrations via South) to create the required database tables
4. Add an entry to your `urls.py`:

```
    urlpatterns += patterns('',
        url(r'^captcha/', include('captcha.urls')),
    )
```

Django-simple-captcha 0.4.3 and later supports both Django 1.7's new migrations and South migrations: if you are using South and Django < 1.7, you must define the following in your settings:

```
SOUTH_MIGRATION_MODULES = {
    'captcha': 'captcha.south_migrations',
}
```

4.1.2 Adding to a Form

Using a `CaptchaField` is quite straight-forward:

Define the Form

To embed a CAPTCHA in your forms, simply add a `CaptchaField` to the form definition:

```
from django import forms
from captcha.fields import CaptchaField

class CaptchaTestForm(forms.Form):
    myfield = AnyOtherField()
    captcha = CaptchaField()
```

. . . or, as a `ModelForm`:

```
from django import forms
from captcha.fields import CaptchaField

class CaptchaTestModelForm(forms.ModelForm):
    captcha = CaptchaField()
    class Meta:
        model = MyModel
```

Validate the Form

In your view, validate the form as usually: if the user didn't provide a valid response to the CAPTCHA challenge, the form will raise a `ValidationError`:

```
def some_view(request):
    if request.POST:
        form = CaptchaTestForm(request.POST)

        # Validate the form: the captcha field will automatically
        # check the input
        if form.is_valid():
            human = True
    else:
        form = CaptchaTestForm()

    return render_to_response('template.html',locals())
```

Passing arguments to the field

`CaptchaField` takes a few optional arguements:

- `output_format` will let you format the layout of the rendered field. Defaults to the value defined in : *CAPTCHA_OUTPUT_FORMAT*.
- `id_prefix` Optional prefix that will be added to the ID attribute in the generated fields and labels, to be used when e.g. several Captcha fields are being displayed on a same page. (added in version 0.4.4)

Example usage for ajax form

An example CAPTCHA validation in AJAX:

```
from django.views.generic.edit import CreateView
from captcha.models import CaptchaStore
from captcha.helpers import captcha_image_url
from django.http import HttpResponse
import json

class AjaxExampleForm(CreateView):
    template_name = ''
    form_class = AjaxForm

    def form_invalid(self, form):
        if self.request.is_ajax():
            to_json_response = dict()
            to_json_response['status'] = 0
            to_json_response['form_errors'] = form.errors

            to_json_response['new_cptch_key'] = CaptchaStore.generate_key()
            to_json_response['new_cptch_image'] = captcha_image_url(to_json_response['new_cptch_key'

            return HttpResponse(json.dumps(to_json_response), content_type='application/json')

    def form_valid(self, form):
        form.save()
        if self.request.is_ajax():
            to_json_response = dict()
            to_json_response['status'] = 1

            to_json_response['new_cptch_key'] = CaptchaStore.generate_key()
            to_json_response['new_cptch_image'] = captcha_image_url(to_json_response['new_cptch_key'

            return HttpResponse(json.dumps(to_json_response), content_type='application/json')
```

And in javascript your must update the image and hidden input in form

Example usage ajax refresh button

# html:

```
<form action='.' method='POST'>
    {{ form }}
    <input type="submit" />
    <button class='js-captcha-refresh'></button>
</form>
```

# javascript:

```
$('.js-captcha-refresh').click(function(){
    $form = $(this).parents('form');

    $.getJSON($(this).data('url'), {}, function(json) {
        // This should update your captcha image src and captcha hidden input
    });

    return false;
});
```

4.2 Advanced topics

4.2.1 Configuration toggles

The following configuration elements can be defined (in your `settings.py`)

CAPTCHA_FONT_PATH

Full path and filename of a TrueType (TTF), OpenType, or pilfont font file used to render text.

Defaults to: `fonts/Vera.ttf` (included in the application, GPL font).

Note that your PIL installation must support TTF and/or OpenFont if you want to use these kind of glyphs (most modern distributions of PIL do.)

Note: as of version 0.4.6, `CAPTCHA_FONT_PATH` may be an iterable of font paths, in which case a font will be picked randomly from the list for each CAPTCHA.

CAPTCHA_FONT_SIZE

Font-size in pixels of the rendered text.

Defaults to '22'.

CAPTCHA_IMAGE_SIZE

Image size in pixels of generated captcha, specified by 2-tuple (width, height)

Defaults to *None* (automatically calculated)

CAPTCHA_LETTER_ROTATION

A random rotation in this interval is applied to each letter in the challenge text.

Defaults to `(-35,35)`.

New in version 0.1.6: set this to None to disable letter roation.

CAPTCHA_BACKGROUND_COLOR

Background-color of the captcha. Can be expressed as html-style #rrggbb, rgb(red, green, blue), or common html names (e.g. "red").

Defaults to: `'#ffffff'`

CAPTCHA_FOREGROUND_COLOR

Foreground-color of the captcha.

Defaults to `'#001100'`

CAPTCHA_CHALLENGE_FUNCT

String representing a python callable (i.e. a function) to use as challenge generator.

See Generators below for a list of available generators and a guide on how to write your own.

Defaults to: `'captcha.helpers.random_char_challenge'`

CAPTCHA_NOISE_FUNCTIONS

List of strings of python callables that take a PIL `DrawImage` object and an `Image` image as input, modify the `DrawImage`, then return it.

Defaults to: `('captcha.helpers.noise_arcs','captcha.helpers.noise_dots',)`

A null noise helper function useful when debugging issues is available at `'captcha.helpers.noise_null'`.

CAPTCHA_FILTER_FUNCTIONS

List of strings of python callables that take a PIL `Image` object as input, modify it and return it.

These are called right before the rendering, i.e. after the noise functions.

Defaults to: `('captcha.helpers.post_smooth',)`

CAPTCHA_WORDS_DICTIONARY

Required for the `word_challenge` challenge function only. Points a file containing a list of words, one per line.

Defaults to: `'/usr/share/dict/words'`

CAPTCHA_FLITE_PATH

Full path to the `flite` executable. When defined, will automatically add audio output to the captcha.

Defaults to: `None` (no audio output)

CAPTCHA_TIMEOUT

Integer. Lifespan, in minutes, of the generated captcha.

Defaults to: 5

CAPTCHA_LENGTH

Sets the length, in chars, of the generated captcha. (for the `'captcha.helpers.random_char_challenge'` challenge)

Defaults to: 4

CAPTCHA_DICTIONARY_MIN_LENGTH

When using the word_challenge challenge function, controls the minimum length of the words to be randomly picked from the dictionary file.

Defaults to: 0

CAPTCHA_DICTIONARY_MAX_LENGTH

When using the word_challenge challenge function, controls the maximal length of the words to be randomly picked from the dictionary file.

Defaults to: 99

Note: it's perfectly safe to specify e.g. `CAPTCHA_DICTIONARY_MIN_LENGTH = CAPTCHA_DICTIONARY_MAX_LENGTH = 6` but it's considered an error to define `CAPTCHA_DICTIONARY_MAX_LENGTH` to be smaller than `CAPTCHA_DICTIONARY_MIN_LENGTH`.

CAPTCHA_OUTPUT_FORMAT

New in version 0.1.6

Specify your own output format for the generated markup, when e.g. you want to position the captcha image relative to the text field in your form.

Defaults to: `u'%(image)s %(hidden_field)s %(text_field)s'`

Note: the three keys have to be present in the format string or an error will be thrown at runtime.

CAPTCHA_TEST_MODE

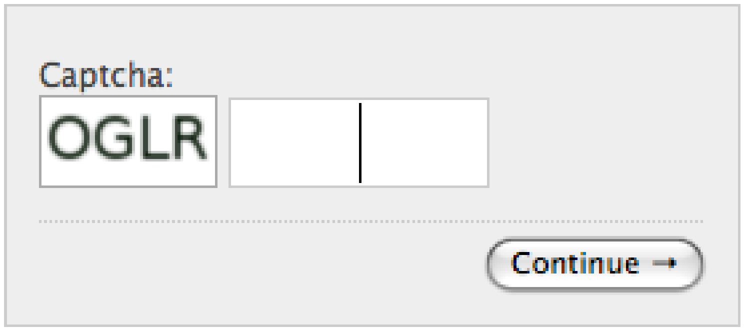
New in version 0.3.6

When set to True, the string "PASSED" (any case) will be accepted as a valid response to any CAPTCHA. Use this for testing purposes. Warning: do NOT set this to True in production.

Defaults to: False

4.2.2 Generators and modifiers

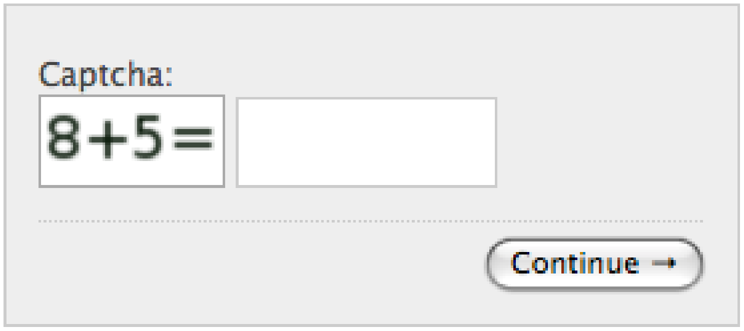
Random chars



Classic captcha that picks four random chars. This is case insensitive.

```
CAPTCHA_CHALLENGE_FUNCT = 'captcha.helpers.random_char_challenge'
```

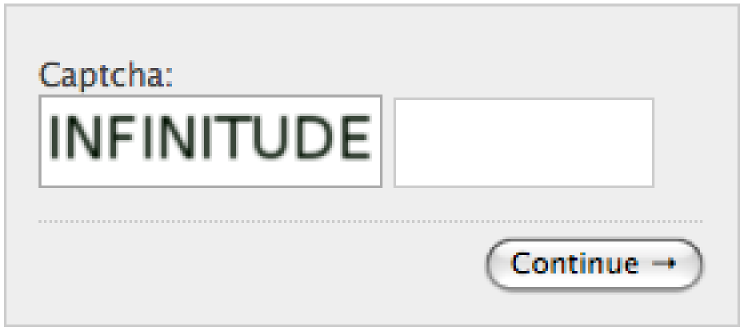
Simple Math



Another classic, that challenges the user to resolve a simple math challenge by randomly picking two numbers between one and nine, and a random operator among plus, minus, times.

```
CAPTCHA_CHALLENGE_FUNCT = 'captcha.helpers.math_challenge'
```

Dictionary Word



Picks a random word from a dictionary file. Note, you must define `CAPTCHA_WORDS_DICTIONARY` in your cofiguration to use this generator.

```
CAPTCHA_CHALLENGE_FUNCT = 'captcha.helpers.word_challenge'
```

Roll your own

To have your own challenge generator, simply point `CAPTCHA_CHALLENGE_FUNCT` to a function that returns a tuple of strings: the first one (the challenge) will be rendered in the captcha, the second is the valid response to the challenge, e.g. `('5+10=', '15')`, `('AAAA', 'aaaa')`

This sample generator that returns six random digits:

```
import random

def random_digit_challenge():
    ret = u''
    for i in range(6):
        ret += str(random.randint(0,9))
    return ret, ret
```