Django Simple Captcha Documentation

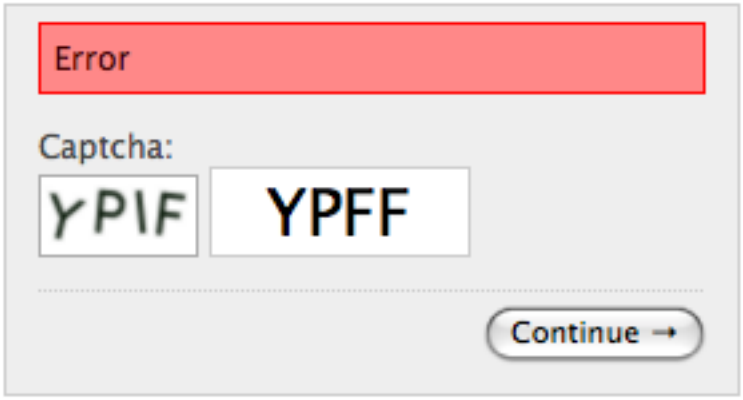
Release 0.5.10

Marco Bonetti

Jan 18, 2019

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 1.8+
- A recent version of Pillow 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.

CHAPTER 4

Contents:

4.1 Using django-simple-captcha

4.1.1 Installation

1. Install `django-simple-captcha` via pip: `pip install django-simple-captcha`
2. Add `captcha` to the `INSTALLED_APPS` in your `settings.py`
3. Run `python manage.py migrate`
4. Add an entry to your `urls.py`:

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

Note: PIL and Pillow require that image libraries are installed on your system. On e.g. Debian or Ubuntu, you'd need these packages to compile and install Pillow:

```
apt-get -y install libz-dev libjpeg-dev libfreetype6-dev python-dev
```

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 usual. 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)
- generator Optional callable or module path to callable that will be used to generate the challenge and the response, e.g. generator='path.to.generator_function' or generator=lambda: ('LOL', 'LOL'), see also *Generators and modifiers*. Defaults to whatever is defined in settings.CAPTCHA_CHALLENGE_FUNCT.

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
    });
```

(continues on next page)

(continued from previous page)

```
    return false;
});
```

Example usage ajax refresh

javascript:

```
$('.captcha').click(function () {
    $.getJSON("/captcha/refresh/", function (result) {
        $('.captcha').attr('src', result['image_url']);
        $('#id_captcha_0').val(result['key'])
    });


});
```

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_MATH_CHALLENGE_OPERATOR

When using the `math_challenge`, lets you choose the multiplication operator. Use lowercase `'x'` for cross sign.

Defaults to: `'*'` (asterisk sign)

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_SOX_PATH

Full path to the `sox` executable. If audio output is enabled via `CAPTCHA_FLITE_PATH`, the generated output audio file is identical across multiple generations (unlike CAPTCHA images which get different random noise each time they are rendered). User appleorange1 has shown that this could be used to pre-generate a "rainbow-table" of all possible input strings and a hash of the generated output soundfile, thus rendering an attack on audio CAPTCHAs trivial.

If sox is installed and used via this settings, random brown noise is injected into the generated audio file, rendering attacks via a rainbow table impossible.

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: `None`

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

Warning: This setting is deprecated in favor of template-based widget rendering (see the Rendering section below).

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

CAPTCHA_GET_FROM_POOL

By default, *django-simple-captcha* generates a new captcha when needed and stores it in the database. This occurs in a *HTTP GET request*, which may not be wished. This default behavior may also conflict with a load balanced infrastructure, where there is more than one database to read data from. If this setting is *True*, when a new captcha is needed, a random one will be just read from a pool of captchas saved previously in the database. In this case, the custom management command *captcha_create_pool* must be run regularly in intervals slightly shorter than *CAPTCHA_TIMEOUT*. A good value for *CAPTCHA_TIMEOUT* could be 1446 (24 hours and 6 minutes) when adding captchas to the pool every 24 hours, and setting *CAPTCHA_GET_FROM_POOL_TIMEOUT* (see below) to 5 minutes. This means that 6 minutes before the last captchas expires, new captchas will be created, and no captcha will be used whose expiration is less than 5 minutes. In this case, use a cronjob or similar to run *python manage.py captcha_create_pool* every 24 hours.

Defaults to: False

CAPTCHA_GET_FROM_POOL_TIMEOUT

This is a timeout value in minutes used only if *CAPTCHA_GET_FROM_POOL* (see above) is *True*. When picking up randomly from the pool, this setting will prevent to pick up a captcha that expires sooner than *CAPTCHA_GET_FROM_POOL_TIMEOUT*.

Defaults to: 5

4.2.2 Rendering

`CaptchaTextInput` supports the widget rendering using template introduced in Django 1.11. To change the output HTML, change the `template_name` to a custom template or modify `get_context` method to provide further context. See https://docs.djangoproject.com/en/dev/ref/forms/renderers/ for description of rendering API. Keep in mind that `CaptchaTextInput` is a subclass of `MultiWidget` which affects the context, see https://docs.djangoproject.com/en/2.0/ref/forms/widgets/#multiwidget.

For example, you would:

```
class CustomCaptchaTextInput(CaptchaTextInput):
    template_name = 'custom_field.html'

class CaptchaForm(forms.Form):
    captcha = CaptchaField(widget=CustomCaptchaTextInput)
```

And then have a `custom_field.html` template:

```
{% load i18n %}
{% spaceless %}
<div class="form-group">
  <label class="control-label">{{ label }}</label>
  <div class="form-group">
    <div class="input-group mb-3">
      <div class="input-group-prepend">
        {% if audio %}
            <a title="{% trans "Play CAPTCHA as audio file" %}" href="{{ audio }}">
        {% endif %}
        <img src="{{ image }}" alt="captcha" class="captcha" />
      </div>
      {% include "django/forms/widgets/multiwidget.html" %}
    </div>
  </div>
</div>
{% endspaceless %}
```

Note: For this to work, you MUST add `django.forms` to your `INSTALLED_APPS` and set `FORM_RENDERER = 'django.forms.renderers.TemplatesSetting'` to your settings.py. (See here for an explanation)

Warning: To provide backwards compatibility, the old style rendering has priority over the widget templates. If the `CAPTCHA_FIELD_TEMPLATE` or `CAPTCHA_OUTPUT_FORMAT` settings or `field_templates` or `output_format` parameter are set, the direct rendering gets higher priority. If widget templates are ignored, make sure you're using Django >= 1.11 and disable these settings and parameters.

Old style rendering

Warning: This rendering method is deprecated. Use Django >= 1.11 and widgets templates instead.

A CAPTCHA field is made up of three components:

- The actual image that the end user has to copy from
- A text field, that the user has to fill with the content of the image
- A hidden field, containing the database reference of the CAPTCHA (for verification).

These three elements are rendered individually, then assembled into a single bit of HTML.

As of version 0.4.7 you can control how the individual components are rendered, as well as how all components are assembled, by overriding four templates:

- `captcha/image.html` controls the rendering of the image (and optionnally audio) element
- `captcha/text_field.html` controls the rendering of the text field
- `captcha/hidden_field.html` controls the rendering of the hidden input
- `captcha/field.html` controls the assembling of the previous three elements

These templates can be overriden in your own `templates` folder, or you can change the actual template names by settings `CAPTCHA_IMAGE_TEMPLATE`, `CAPTCHA_TEXT_FIELD_TEMPLATE`, `CAPTCHA_HIDDEN_FIELD_TEMPLATE` and `CAPTCHA_FIELD_TEMPLATE`, respectively.

Context

The following context variables are passed to the three "individual" templates:

- `image`: The URL of the rendered CAPTCHA image
- `name`: name of the field (i.e. the name of your form field)
- `key`: the hashed value (identifier) of this CAPTCHA: this is stored and passed in the hidden input
- `id`: the HTML `id` attribute to be used

The `captcha/field.html` template receives the following context:

- `image`: the rendered (HTML) image and optionnaly audio elements
- `hidden_field`: the rendered hidden input
- `text_field`: the rendered text input

Note: these elements have been marked as safe, you can render them straight into your template.

4.2.3 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 configuration 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
```

4.3 Version History

4.3.1 Version 0.5.10

- Test against Django 2.2a1
- Docs: Grammar correction (#160, thanks @DanAtShenTech)
- Fix: Add '+' to text replacement for audio support (#157, thanks @geirkairam)
- I18N: Added Swedish translation (#155, thanks @stefannorman)
- Docs: Provide an example of custom field template (#158, thanks @TheBuky)

4.3.2 Version 0.5.9

- Add missing Jinja2 templates in the pypi packages.

4.3.3 Version 0.5.8

- Add support for Jinja2 templates (Issue #145, PR #146, thanks @ziima)
- Cleanup, drop dependency on South (#141, #142 thanks @ziima)

4.3.4 Version 0.5.7

- Use templates for rendering of widgets (Issue #128, #134, PR #133, #139, thanks @ziima)
- Always defined audio context variable (PR #132, thanks @ziima)
- Test against Django 2.1a
- Updated AJAX update docs (PR #140, thanks @CNmanyue)
- Fixed a typo in a variable name (PR #130, thanks @galeo)

4.3.5 Version 0.5.6

- Updated render method to adapt for Django 2.1 (PR #120, thanks @skozan)
- Improved compatibility with Django 2.0, tests against Django 2.0a1 (PR #121, thanks @Kondou-ger)
- Dropped support for PIL (use Pillow instead)
- Updated documentation (Fixes #122, thanks @claudep)
- Test against Django 2.0b1
- Return a Ranged Response when returning WAV audio to support Safari (Fixes #123, thanks @po5i)
- Optionally inject brown noise into the generated WAV audio file, to avoid rainbow-table attacks (Fixes #124, thanks @appleorange1)
- Test against Django 2.0

4.3.6 Version 0.5.5

- I messed the 0.5.4 release, re-releasing as 0.5.5

4.3.7 Version 0.5.4

- Removed a couple gremlins (PR #113, thanks @Pawamoy)
- Added autocapitalize="off", autocorrect="off" and spellcheck="false" to the genreated field (PR #116, thanks @rdonnelly)
- Test against Django 1.11
- Drop support of Django 1.7 ("it'll probably still work")

4.3.8 Version 0.5.3

- Ability to pass a per-field challenge generator function (Fixes #109)
- Added a feature to get captchas from a data pool of pre-created captchas (PR #110, thanks @skozan)
- Cleanup to remove old code handling timezones for no longer supported Django versions
- Fix for "Size must be a tuple" issue with Pillow 3.4.0 (Fixes #111)

4.3.9 Version 0.5.2

- Use any mutliplication uperator instead of "*". (Fixes #77 via PR #104, thanks @honsdomi and @isergey)
- Test against Django 1.10

4.3.10 Version 0.5.1

- Fine tuning MANIFEST.in
- Prevent testproject from installing into site-packages

4.3.11 Version 0.5.0

- Adds missing includes in MANIFEST.in

4.3.12 Version 0.4.7

- Supported Django versions are now 1.7, 1.8 and 1.9
- Trying to fix the TravisCI build errors
- Use Django templates to render the individual fields, as well as the assembled Captcha Field (Issue #31)

4.3.13 Version 0.4.6

- Fixes an UnicodeDecodeError which was apparently only triggered during testing on TravisCI (I hope)
- Support for Django 2.0 urlpatterns syntax (PR #82, Thanks @R3v1L)
- settings.CAPTCHA_FONT_PATH may be a list, in which case a font is picked randomly (Issue #51 fixed in PR #88, Thanks @inflrscns)

4.3.14 Version 0.4.5

- Test with tox
- Test against Django 1.8 final
- Added ability to force a fixed image size (PR #76, Thanks @superqwer)

4.3.15 Version 0.4.4

- Added id_prefix argument (fixes issue #37)

4.3.16 Version 0.4.3

- Add null noise helper (Thanks @xrmx)
- Test against Django 1.7b4
- Added Spanish translations (Thanks @dragosdobrota)
- Massive cleanup (pep8, translations)
- Support for transparent background color. (Thanks @curaloucura)
- Support both Django 1.7 migrations and South migrations. Please note, you *must* add the following to your settings, if you are using South migrations and Django 1.6 or lower.
- Make sure autocomplete="off" is only applied to the text input, not the hidden input (Issue #68, thanks @narrowfail)
- Fixed some grammar in the documentation. (Thanks @rikrian)
- Return an HTTP 410 GONE error code for expired captcha images, to avoid crawlers from trying to reindex them (PR #70, thanks @joshuajonah)
- Fixed title markup in documentation (#74, thanks @pavlov99)
- Test against Django 1.7.1

4.3.17 Version 0.4.2

- Added autocomplete="off" to the input (Issue #57, thanks @Vincent-Vega)
- Fixed the format (msgfmt -c) of most PO and MO files distributed with the project
- Added Bulgarian translations. (Thanks @vstoykov)
- Added Japanese translations. (Thanks, Keisuke URAGO)
- Added Ukrainian translations. (Thanks, @FuriousCoder)

- Added support for Python 3.2. (Thanks, @amrhassan)

4.3.18 Version 0.4.1

- Dropped support for Django 1.3
- Fixed support of newer versions of Pillow (2.1 and above. Pillow 2.2.2 is now required) Thanks @viaregio (Issue #50)

4.3.19 Version 0.4.0

- Perfom some tests at package installation, to check whether PIL or Pillow are already installed. (Issue #46)
- Added Slovak translations. (Thanks @ciklysta)

4.3.20 Version 0.3.9

- Run most tests both with a regular Form and a ModelForm, to avoid regressions such as Issue #40
- Handle the special case where CaptchaFields are instantiated with required=False (Issue #42, thanks @DrMeers)
- Fixed a misspelled setting, we now support both spellings, but the docs suggest the correct one (Issue #36, thanks @sayadn)
- Added Django 1.6b to testrunner and adapted the test cases to support Django 1.6's new test discovery
- Added German translations. (Thanks @digi604)
- Frozen the version of Pillow to 2.0.0, as 2.1.0 seems to be truncating the output image – Issue #44, Thanks @andruby
- Added Polish translations. (Thanks @stilzdev)

4.3.21 Version 0.3.8

- Fixed a critical bug (Issue #40) that would generate two captcha objects, and the test would always fail. Thanks @pengqi for the heads-up.

4.3.22 Version 0.3.7

- Improved Django 1.5 and Django HEAD (1.6) compatibility (thanks @uruz)
- Python3 compatibility (requires six and Pillow >= 2.0)
- Added zh_CN localization (thanks @mingchen)
- Make sure the generated challenge is a string type (the math challenge was probably broken – Issue #33, thanks @YDS19872712)
- Massive cleanup and refactoring (Issue #38, thanks @tepez)
- Test refactoring to test a couple generators that weren't tested by default

4.3.23 Version 0.3.6

- Django 1.5 compatibility (only affects tests)
- Italian localization (thanks @arjunadeltoso)
- Russian localization (thanks @mikek)
- Fixed issue #17 - Append content-length to response (thanks @shchemelevev)
- Merged PR #19 - AJAX refresh of captcha (thanks @artofhuman)
- Merged PR #22 - Use op.popen instead of subprocess.call to generate the audio CAPTCHA (thanks @beda42)
- Fixed issue #10 - uniformize spelling of "CAPTCHA" (thanks @mikek)
- Fixed issue #12 - Raise error when try to initialize CaptchaTextInput alone and/or when try to initialize CaptchaField with widget keyword argument (thanks @vstoykov)
- Merged PR #15 - Allow a 'test mode' where the string 'PASSED' always validates the CAPTCHA (thanks @beda42)
- Dutch translation (thanks @leonderijke)
- Turkish translation (thanks @gkmngrgn)

4.3.24 Version 0.3.5

- Fixes issue #4: Fixes id_for_label malfunction with prefixed forms (thanks @lolek09)

4.3.25 Version 0.3.4

- Fixes issue #3: regression on Django 1.4 when USE_TZ is False

4.3.26 Version 0.3.3

- Django 1.4 Time zones compatibility
- PEP 8 love

4.3.27 Version 0.3.2

- Added a test project to run tests
- Added South migrations