# Django REST framework reCAPTCHA Documentation

*Release 0.2.0*

Maximilien Raulic

Feb 24, 2019

# Contents

Django REST framework reCAPTCHA provides you a serializer field to handle and validate Google reCAPTCHA response.

See our *Changelog* for information on updates.

# CHAPTER 1

# Requirements

- Python: 2.7, 3.4, 3.5, 3.6, 3.7
- Django: 1.10, 1.11, 2.0, 2.1
- Django REST framework: 3.4, 3.5, 3.6, 3.7, 3.8, 3.9

# CHAPTER 2

# Sources

`djangorestframework-recaptcha` is hosted on Github.

# CHAPTER 3

# Index

## 3.1 Installation

To install Django REST framework reCAPTCHA, run this command in your terminal:

```
$ pip install djangorestframework-recaptcha
```

This is the preferred method to install Django REST framework reCAPTCHA, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

Once the `djangorestframework-recaptcha` installed, add it to your `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    ...
    "rest_framework_recaptcha",
    ...
)
```

Next, register yourself and obtain your reCAPTCHA credentials at https://www.google.com/recaptcha/admin.

Finally, copy/paste your Google reCAPTCHA secret key to the `DRF_RECAPTCHA_SECRET_KEY` setting:

```
DRF_RECAPTCHA_SECRET_KEY = "<your_reCAPTCHA_secret_key>"
```

## 3.2 Settings

### 3.2.1 `DRF_RECAPTCHA_VERIFY_ENDPOINT`

API endpoint to which to send the information to validate the reCAPTCHA response token. The default value is *https://www.google.com/recaptcha/api/siteverify* (cf. reCAPTCHA documentation).

### 3.2.2 DRF_RECAPTCHA_SECRET_KEY

Secret key of your reCAPTCHA application. Don't forget to fill in this settings with your reCAPTCHA application secret key.

## 3.3 Usage

To use Django REST framework reCAPTCHA within your project you'll need to import and add the `ReCaptchaField` serializer field into the wanted serializer. For example:

```
from rest_framework import serializers
from rest_framework_recaptcha import ReCaptchaField


class MySerializer(serializers.Serializer):
    recaptcha = ReCaptchaField()
```

For you information, `ReCaptchaField` fields are defined as `write_only=True` by default.

Once your serializer is configured with your `ReCaptchaField` you'll be able to send the client side generated reCAPTCHA response token and validate it server side (cf. reCAPTCHA documentation).

## 3.4 Validation errors

During the validation process of the reCAPTCHA response token by the verification API and according to the documentation, the following errors may be raised:

| Error code | Message |
|---|---|
| missing-input-secret | The secret parameter is missing. |
| invalid-input-secret | The secret parameter is invalid or malformed. |
| missing-input-response | The response parameter is missing. |
| invalid-input-response | The response parameter is invalid or malformed. |
| bad-request | The request is invalid or malformed. |
| timeout-or-duplicate | The response parameter has timed out or has already been used. |

Each of these errors are handled by the `ReCaptchaValidator`. In case of an unknown error the `bad-request` error will be raised.

Each error message can be replaced if needed. For this, you have two options:

1. Create a custom `ReCaptchaField` that inherits from `ReCaptchaField` while redefining the `default_error_messages` attribute with a dictionary which for each entry the key will match the code to override and the value to the new message. Example:

```
from rest_framework import serializers
from rest_framework_recaptcha import ReCaptchaField


class MyReCaptchaField(ReCaptchaField):
    default_error_messages = {
        "invalid-input-response": "reCAPTCHA token is invalid.",
    }
```

(continues on next page)

(continued from previous page)

```


class MySerializer(serializers.Serializer):
    recaptcha = MyReCaptchaField()
```

2. When adding the `ReCaptchaField` field to your serializer you can pass the optional `error_messages` parameter with a dictionary which for each entry the key will match the code to override and the value to the new message. Example:

```
from rest_framework import serializers
from rest_framework_recaptcha import ReCaptchaField


class MySerializer(serializers.Serializer):
    recaptcha = ReCaptchaField(
        error_messages={
            "invalid-input-response": "reCAPTCHA token is invalid.",
        }
    )
```

## 3.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

### 3.5.1 Types of Contributions

#### Report Bugs

Report bugs at https://github.com/Maximilien-R/django-rest-framework-recaptcha/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

#### Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

#### Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

#### Write Documentation

Django REST framework reCAPTCHA could always use more documentation, whether as part of the official Django REST framework reCAPTCHA docs, in docstrings, or even on the web in blog posts, articles, and such.

#### Submit Feedback

The best way to send feedback is to file an issue at https://github.com/Maximilien-R/django-rest-framework-recaptcha/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

### 3.5.2 Get Started!

Ready to contribute? The easiest way to work on `djangorestframework-recaptcha` for local development is to use Docker. `djangorestframework-recaptcha` come with a `Dockerfile` and a `Makefile` that implements all the needed stuff and helpers to work on this project.

1. Install Docker.
2. Fork the `djangorestframework-recaptcha` GitHub repository.
3. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-rest-framework-recaptcha.git
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

Now you can make your changes locally.

5. When you're done making changes, check that your changes pass format check, flake8, complexity and the tests, including testing other Python versions with tox:

```
$ make check-format
$ make style
$ make complexity
$ make test-all
```

6. Format your code and then commit your changes and push your branch to GitHub:

```
$ make format
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

### 3.5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring.
3. The pull request should work for Python 2.7, 3.4, 3.5, 3.6 and 3.7, and for PyPy. Check https://travis-ci.org/Maximilien-R/django-rest-framework-recaptcha/pull_requests and make sure that the tests pass for all supported Python versions.
4. Your code need to be formatted. On this project we use the black code formatter. You can easily format your code with this command: `make format`.

### 3.5.4 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in `HISTORY.rst`). Then run:

```
$ make bumpversion -e VERSION_PART=patch # options: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

### 3.5.5 Translations

You can also participate in the project by adding new language or improving translations.

To add a new language:

```
$ make build-translations -e LOCALE=en
```

To update available languages and check for new strings:

```
$ make update-translations
```

To compile translations:

```
$ make compile-translations
```

## 3.6 Credits

### 3.6.1 Development Lead

- Maximilien Raulic <maximilien.raulic@gmail.com>

### 3.6.2 Contributors

None yet. Why not be the first?

## 3.7 History

### 3.7.1 [Unreleased]

#### Added

- Update dependency packages versions.
- Formats Python imports with `isort`.
- Add SAST through `bandit`.
- Add dependency scan through `safety`.
- Deployment automation of an alpha package version on test PyPI for each branch (except `master`).
- Deployment automation of an production package version on test PyPI for `master` branch.

### 3.7.2 0.2.0 (2018-12-21)

#### Added

- Django REST framework 3.9, Python 3.7 & Django 2.1.
- Set long description content type to reStructuredText.

### 3.7.3 0.1.0 (2018-07-02)

#### Added

- First release on PyPI.