formsender Documentation

Release 0.1

OSUOSL

September 03, 2015

Contents

Contents:

CHAPTER 1

Information

The formsender app is a simple program that is designed to aid in the collection of form data. When a form is submitted, the app gathers the data from it and emails it to a configured email address.

CHAPTER 2

How To Use Formsender

In conf.py.dist there are several settings that Formsender relies on. To use these settings copy them to a new conf.py file in the root directory. You can change the following variables to match your setup:

```
EMAIL = u'email@example.com'
TOKN = u's0m3T0k3n$tr1ng'
CEILING = 10
DUPLICATE_CHECK_TIME = 3600
HOST = "0.0.0.0"
PORT = 5000
SMTP_HOST = "smtp.osuosl.org"
FROM = "formsender@osuosl.org"
```

- `EMAIL` is where the form data will be sent.
- `TOKN` is the validating token from the form. This must match a hidden field in your form called 'tokn'. You can find and set the `TOKN` variable in your conf.py file. Just make sure you also set the hidden `tokn` field value to match.
- `CEILING` is the maximum number of submit requests formsender will accept per second.
- `DUPLICATE_CHECK_TIME` is the time (in seconds) to check past form submissions for duplicate submissions.
- `HOST` and `PORT` is where the `run_simple` listener listens for POST requests
- `SMTP_HOST` sets the host for the `sendmail` function. Must be a smtp server
- `FROM` is the address the email will be sent from

You can run flake8 on request_handler (the application):

```
$ make flake
```

Once a valid conf.py file exists, tests can be run:

```
$ make tests
```

To run the application locally for development purposes:

```
$ make run
```

The app will now wait at `HOST:PORT` for the form to be submitted, and will email the information submitted to the email specified. `HOST` and `PORT` can be changed in conf.py to match your desired setup.

CHAPTER 3

Set Up a Form

To use Formsender, you need to write an html form with several required and optional fields. Formsender uses these fields to authenticate the form and format the outgoing email message. To include these fields in your form just set the `name`, `type`, and `value` properties like this:

```
<input type="hidden" name="last_name" value=""/>
```

3.1 Required Fields

To use Formsender, you need to write an html form with several required fields. Formsender uses these fields to attempt to authenticate the form (make sure it is not from a robot).

Include required fields by setting the `name` property to the following:

- **email** - must contain a valid email on submission

  example: `<input type="text" name="email" value="" size="60" maxlength="128" />`

- **name** - cannot be empty on submission

  example: `<input type="text" name="name" value="" size="60" maxlength="128" />`

- **last_name** - not for an actual last name field. Must be empty, must be hidden

  example: `<input type="hidden" name="last_name" value=""/>`

- **tokn** - contents must match TOKN in conf.py, must be hidden

  example: `<input type="hidden" name="tokn" value="s0m3T0k3n$tr1ng" />`

- **redirect** - url to redirect to on form submission, if an error occurs a query string will be added with an error message. Should be hidden.

  example: `<input type="hidden" name="redirect" value="http://www.example.com" />`

3.2 Optional Fields

Formsender uses an additional optional field to help format your outgoing email:

- **mail_subject**

sets outgoing email subject to mail_subject's contents. If mail_subject is not included, the subject will default to `Form Submission`. This should be a hidden field.

example: `<input type="hidden" name="mail_subject" value="FORM: New Test Submission" />`

CHAPTER 4

All Other Fields

Formsender formats the email like so:

```
Contact:
--------
NAME:   Submitted Name
EMAIL:   email@example.com

Information:
------------
Community Size:

About 15 developers

Deployment Timeframe:

Within 7 days

Distribution:

Fedora

Duration Of Need:

Six months
```

The contact information, name and email, is placed at the beginning of the email. All following fields are placed in alphabetical order by the input `name`. Formsender formats each input `name` to title case and uses it as titles in the email. **Make sure these name fields are descriptive** and do not use strange formatting like the following:

```
<input type="text" name="submitted[distribution]" value="" />
```

Formsender does not know how to interpret this name and will result in a `Bad Request` error from the server.

CHAPTER 5

Error Codes

Formsender returns different error codes when invalid data is sent from the form.

Error Number	Error Message	Cause
1	Invalid Email	User submitted an invalid email
2	Invalid Name	Name field was empty
3	Improper Form Submission	Hidden field was not empty or token was invalid
4	Too Many Requests	Number of submissions violated CEILING variable from conf.py
5	Duplicate Request	This request is a duplicate of an earlier request

These error codes can be handled with a little javascript in your redirect page:

```
// Get error number and message from query string
function getQueryVariable(variable) {
  var query = window.location.search.substring(1);
  var vars = query.split("&");
  for (var i = 0; i < vars.length; i++) {
    var pair = vars[i].split("=");
    if(pair[0] == variable) {
      return pair[1];
    }
  }
  return (false);
}

var errorNumber = getQueryVariable("error");
var errorMessage = getQueryVariable("message");

// errorMessage will only be a string if a query string is present.
// If a query string is present, there was an error. Format the message.
if (typeof errorMessage == "string") {
  errorMessage = errorMessage.replace("+", " ").replace("/", "");
}

// If both these exist, there was an error with the submission, write to page
if (errorNumber && errorMessage) {
  document.write("<h3 style='color:red'>An error occurred with your form submission</h3>",
                 "<p style='color:red'>Error number: ", errorNumber, "</p>",
                 "<p style='color:red'>Error message: ", errorMessage, "</p>");
}
```

CHAPTER 6

Using Formsender with Docker

Formsender ships with a Dockerfile for easier development. Consult the docker documentation for instructions on how to use docker: http://docs.docker.com/

6.1 Build the Container

If you haven't made any changes to the source code, build the containers by running:

```
$ docker-compose build
```

If changes have been made, run:

```
$ docker-compose build --no-cache
```

This tells docker to include the updated source code in the build.

6.2 Run Formsender in the Container

Now that the container has been built, you can run the app in the container:

```
$ docker-compose up
```

If the original settings in conf.py, Dockerfile, and docker-compose.yml are unchanged, Formsender will now be running on the docker container's port 5000, which is bound to the host's port 5000. Forms sent to port 5000 on the host will communicate with Formsender correctly.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search