streaming-form-data Documentation

Release 0.4.0

Siddhant Goel

Jan 09, 2018

Contents:

`streaming_form_data` provides a Python parser for parsing `multipart/form-data` input chunks (the most commonly used encoding when submitting values through HTML forms). Chunk size is determined by the API user, but currently there are no restrictions on what the size should be, since the parser works byte-by-byte. This also means that passing the entire input as a single chunk should also work.

Please note, that this library has only been tested with Python 3 (specifically, versions 3.3, 3.4, 3.5, and 3.6). Python 2.7 is not supported yet, but pull requests are always welcome.

CHAPTER 1

Installation

```
$ pip install streaming_form_data
```

The core parser is written in `Cython`, which is a superset of Python but compiles the input down to a C extension which can then be imported in normal Python code.

The compiled C parser code is included in the PyPI package, hence the installation requires a working C compiler.

CHAPTER 2

Usage

```
>>> from streaming_form_data import StreamingFormDataParser
>>> from streaming_form_data.targets import ValueTarget, FileTarget, NullTarget
>>>
>>> headers = {'Content-Type': 'multipart/form-data; boundary=boundary'}
>>>
>>> parser = StreamingFormDataParser(headers=headers)
>>>
>>> parser.register('name', ValueTarget())
>>> parser.register('file', FileTarget('/tmp/file.txt'))
>>> parser.register('discard-me', NullTarget())
>>>
>>> parser.data_received(chunk)
```

The parser is fed chunks of (bytes) input, and takes action depending on what the current byte is. In case it notices input that's expected (input that has been registered by calling `parser.register`, it will pass on the input to the registered `Target` class which will then decide what to do with it. In case there's a part which is not needed, it can be associated to a `NullTarget` object and it will be discarded.

CHAPTER 3

API

The API has two main points of entry:

- `StreamingFormDataParser` class - This is the main entry point, and expects a dictionary of request `headers`. These headers are used to determine the input `Content-Type`.
- `Target` class(es) - When registering inputs with the parser, instances of subclasses of the `Target` class should be used, since these targets ultimately determine what to do with the data.

Currently the following three `Target` classes are included with this library.

- `ValueTarget` - holds the input in memory
- `FileTarget` - pipes the input to a file on disk
- `SHA256Target` - computes the SHA-256 hash of the input

Any new targets should inherit `streaming_form_data.targets.BaseTarget` and define a `data_received` function.

CHAPTER 4

Examples

Most of the testing has been done using the `Tornado` web framework which allows reading HTTP request data as it arrives in chunks. If you'd like to document usage with another web framework (which ideally allows chunked reads), please open an issue or a pull request.

- `Tornado` - https://git.io/vHeqQ

CHAPTER 5

Indices and tables

- search