Stamper Documentation

Release 0.1

Fco. de Borja Lopez Rio, Oscar M. Lage

Sep 06, 2019

Contents

This is the official documentation for Stamper, a *time tracking* tool written in Python.

Contents:

CHAPTER 1

About Stamper

Contents

1.1 From where does it come from?

Stamper was born strongly inspired by the original stamp tool written by Sascha Welter (aka betabug).

The idea is quite simple:

1. You **stamp** a time in a text file, recording that time as the time when you started working on something:

   ```
   2014-08-13 09:45 start
   ```

2. You **stamp** again once you have finished, providing something like an id (to identify a customer, project, whatever) and a description of what you did:

   ```
   2014-08-13 10:45 stamper writing documentation
   ```

Those times are saved into a text file and afterwards, with a little bit of python magic, you get some reports about the amount of time dedicated per day or id (for example).

1.2 Enter the JSON

Betabug's stamp tool records those times as lines in a text file. This is quite convenient because you can look into the file with traditional tools like *cat*, *head* or *tail*, or even open it with your favourite editor, and get information about the times you've spent working on something.

But this comes *at a cost*, it needs some *ugly* parsing of the .txt file in the python code that calculates the statistics and reports, making it difficult/slow to perform certain tasks.

And here it comes JSON:

```
JSON (JavaScript Object Notation) is a lightweight data-interchange format.
It is easy for humans to read and write. It is easy for machines to parse
and generate. It is based on a subset of the JavaScript Programming Language,
Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is
completely language independent but uses conventions that are familiar to
programmers of the C-family of languages, including C, C++, C#, Java,
JavaScript, Perl, Python, and many others. These properties make JSON an
ideal data-interchange language.
```

(extracted from http://json.org)

Using a .json file instead of a txt file, we can export/import python data structures (like lists or dicts) easily into a text-like file. This means that we can still edit the file using a text editor, or check its contents using *cat*, *head* or *tail*, but we can forget about the parsing of the entries.

1.3 The tools

Stamper comes with a variety of tools, but the most used ones are:

- **stamp:** Use this to *stamp* times into the json file.
- **stamps:** USe this to query the json file and obtain information about your stamped times.

See also:

:doc:using

CHAPTER 2

Installation instructions

Installing Stamper is quite easy. If you have some experience installing Python packages[1], you already know how to do it. Stamper is a standard Python package available on pypi[2] so just use your favourite tool (pip, **easy_install_**, etc) to intall it:

```
pip install Stamper
```

You can also grab the latest sources from the project main repository (Mercurial) at https://code.codigo23.net/hg/stamper:

```
hg clone https://code.codigo23.net/hg/stamper
```

And install it from there (development mode):

```
pip install -e /PATH/TO/STAMPER-CLONE
```

[1] http://docs.python.org/tutorial/modules.html#packages
[2] http://pypi.python.org/pypi/stamper

CHAPTER 3

How to use it

Contents

3.1 Recording time

To start recording time:

```
stamp
```

To stop recording time:

```
stamp ID DESCRIPTION
```

where:

- **ID** is a string **without spaces** identifying a customer, project, group of tasks or any other thing you want to use to group your stamps
- **DESCRIPTION** is a description of what you did

for example:

```
stamp stamper writing documentation
```

or:

```
stamp stamper implementing per-month line charts
```

3.1.1 Recording consecutive stamps

Imagine you start working on a project, you *stamp* your start:

```
stamp
```

and then you finish a given task, so you *stamp* it:

```
stamp stamper writing installation instructions
```

but then, you keep yourself working on the same project, but on some other task. You don't have to *stamp* the start time again, you simply *stamp* again when you finish the work on that other task:

```
stamp stamper writing usage instructions
```

Note: This apply also for changing to another project, you don't have to *stamp* the start time again, just keep stamping the end times. In such a situation, Stamper will take the end time of a recorded stamp as the start time of the next one.

3.2 Retrieving information from the existing stamps

3.2.1 Total times

- Get a list of total times stamped per **ID**:

  ```
  stamps
  ```

- Get the total times stamped for a given **ID**, for example, **stamper**:

  ```
  stamps stamper
  ```

- Get the total times stamped for the **ID stamper** during the last week:

  ```
  stamps stamper 1w
  ```

  See also:

  1w is a date-based filter. You can learn more about those filters in the *Date-based filtering* documentation below.

3.2.2 Per day details

- Get a list of detailed per-day times:

```
stamps -v
```

- Get details for a given **ID**, for example, **stamper**:

```
stamps -v stamper
```

- Get details for a given month, for example, *august 2014*:

```
stamps -v 2014-08-01--2014-09-01
```

See also:

2014-08-01–2014-09-01 is a date-based filter. You can learn more about those filters in the *Date-based filtering* documentation below.

3.2.3 Timeline

You can get a *timeline* of all the stamped stamps:

```
stamps -t
```

This *timeline* will show each stamp in a line, in a similar way to what bebu's stamp[1] stores in its text-based file:

```
2014-08-12 12:33 start
2014-08-12 12:51 stamper writing installation instructions
2014-08-12 13:11 stamper writing usage documentation
2014-08-13 09:33 start
2014-08-13 12:12 stamper adding missing tests for the cli tools
```

You can filter the timeline by **ID**:

```
stamps -t stamper
```

And/or by date:

```
stamps -t 2m

.. seealso::

   *2m* is a date-based filter. You can learn more about those filters
   in the :ref:`date_based_filtering` documentation below.
```

3.2.4 Charts

You can generate some nice charts from your stamps. So far only bar charts showing the total time per day can be generated, but you can filter them by **ID** and/or date. Some examples:

- Render a chart of all the times already stamped, by **ID**:

[1] http://repos.betabug.ch/stamp

```
stamps -g
```

- Render a chart of the stamps for the last 4 days:

```
stamps -g 4d
```

- Render a chart of the stamps for the **ID** *stamper* for the last week:

```
stamps -g stamper 1w
```

Note: The charts will be saved into **~/.workstamps-charts**, in SVG format. The name of the chart will be generated based on the current date and time, and a *symbolic link* called **chart-latest.svg** will be created, pointing to the latest chart generated.

3.3 stamps - full List of arguments

You can get the full list of arguments from the command line:

```
stamps -h
```

or:

```
stamps --help
```

3.4 Date-based filtering

Most of the results returned by the *stamps* tool can be filtered using the following date-based filters:

- **YYYY-MM-DD**: Limit the results to only those stamps stamped on this date:

```
stamps -v 2014-08-13
```

```
stamps -t stamper 2014-08-13
```

- **YYYY-MM-DD***: Limit the results to only those stamps stamped **after this date**. For example, this will show per-day details for stamps stamped after the first of august, 2014:

```
stamps -v 2014-08-01*
```

- ***YYYY-MM-DD**: Limit the results to only those stamps stamped **before this date**. For example this will show the total time stamped for project stamper before the first of january, 2014:

```
stamps stamper *2014-01-01
```

- **YYYY-MM-DD–YYYY-MM-DD**: Limit the results to only those stamps stamped in a date **between those dates**. For example, this will show the *timeline* for july, 2014:

```
stamps -t 2014-07-01--2014-08-01
```

- **nD|W|M|Y**: Limit the results to only those stamps stamped since n *days* (**D**), *weeks* (**W**), *months* (**M**) or *years* (**Y**) ago. For example, this will show the details for the last month:

```
stamps -v 1m
```

Note: It does not matter if you provide *d|w|m|y* or *D|W|M|Y*. The filters are handled in a case-insensitive way.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search