tally Documentation

Release 1.0.0

Dan Watson

Sep 27, 2017

Contents

Contents:

CHAPTER 1

Installation

```
pip install tally
```

The tally Server

Tally can be easily added to your existing Django site:

1. Add `tally` to your `INSTALLED_APPS`
2. Include `tally.urls` in your URLconf somewhere, for example:

```
url(r'^_tally/', include('tally.urls'))
```

3. Run `manage.py syncdb` to create tally's Archive table
4. Create one or more Archives in the Django admin (or via the API, coming soon)
5. Run `manage.py tallyserver` to start listening for metrics

Note: Unless you want your metrics to be public, you should take care to protect the URLs using HTTP basic authentication, or some other method. You may also consider running a separate tally instance inside your network that your production site sends metrics to.

Sending Metrics

Use the `tally.tally` method to send a metric for aggregation:

`tally.`**`tally`**(*data*, *value=1*, *timestamp=None*, *host=None*, *port=None*)
Sends a metric to the specified host:port

CHAPTER 2

Configuration

Settings

- `TALLY_DATA_DIR`: A directory where tally will store its archive data (will be created if it does not exist)
- `TALLY_HOST`: The IP address tally should listen on, and the default address for clients to send metrics to (default: 127.0.0.1)
- `TALLY_PORT`: The port tally should use (default: 8900)
- `TALLY_FLUSH_TIME`: The time, in seconds, between flushes of received metric data (default: 5)
- `TALLY_INSTALL_ADMIN`: Whether the Django admin should be included in tally's URLconf. Typically set to True for standalone installations, and False for embedded application installations.
- `TALLY_PREFIX`: If specified, this string will be prepended to every metric the client sends (default: '')

Middleware

Tally includes a simple middleware class, `tally.middleware.PageTimingMiddleware`, that sends the amount of time each request takes to the metrics server. The key is generated by stripping leading and trailing slashes, transforming remaining slashes to periods, and prepending `requests.[method]`. So a POST request for `/myapp/some-page/update/` is recorded as `requests.post.myapp.some-page.update`.

CHAPTER 3

Tally API

The real power of tally lies in its JSON API. Each API endpoint accepts the following GET parameters:

- `q`: a glob pattern (e.g. `requests.get.*`) for matching metric names
- `since`: earliest timestamp to return metrics for
- `until`: latest timestamp to return metrics for
- `pretty`: indicates the JSON should be pretty-printed (useful for debugging or just playing around)

Values

These API endpoints return metrics broken down by both metric name and timestamp.

`/values/<slug>/` Returns a dictionary of metric keys contained in the archived identified by `<slug>`, each mapping to a dictionary of timestamps with all aggregate values.

`/values/<slug>/by<time|name>/` Breaks down the values by time or name. Note that this indicates the "second-level" breakdown, so specifying "byname" will first break down values by time, then by name.

`/values/<slug>/<aggregate>/` Returns the same breakdown as `/values/<slug>/`, but only returns the specified aggregate, instead of all aggregates. `<aggregate>` may be one of: `avg`, `min`, `max`, `sum`, or `count`.

`/values/<slug>/<aggregate>/by<time|name>/` Returns the specified aggreagte, broken down by either time or name.

Aggregates

These API endpoints return metrics broken down only by name or timestamp, then aggregated. So if broken down by name (for instance), instead of returning metrics for every timestamp, the metrics are appropriately aggregated (summing `count` and `sum`, averaging `avg`, etc.) and one value is returned per name.

/aggregate/<slug>/ Returns aggregated metrics for each timestamp. Equivalent to `/aggregate/<slug>/bytime/`.

/aggregate/<slug>/by<time|name>/ Returns aggregated metrics for names or timestamps.

/aggregate/<slug>/<aggregate>/ Returns only the specifed aggregate per timestamp.

/aggregate/<slug>/<aggregate>/by<time|name>/ Returns only the specified aggregate per name or timestamp.

Useful Examples

Imagine an archive named "daily" that records page request timings with a 10 second resolution and 24 hour retention, via the `PageTimingMiddleware` (see *Middleware*):

/aggregate/daily/avg/ This will return all timestamps for the last 24 hours (every 10 seconds), along with the average request time for each 10 second window.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index

T