go-jsbox-metrics-helper Documentation

Release 0.1.1

Praekelt Foundation

November 10, 2014

Contents

Contents:

CHAPTER 1

Metrics Helper

class `MetricsHelper`()
A helper for common metrics tasks.

Arguments

- **im** (*InteractionMachine*) – The interaction machine that the metrics should be run on.

`MetricsHelper.add.`**`sessions_between_states`**(*state_from*, *state_to*, *label*)
Adds an average metric that measures the amount of sessions between two state events. The metric will fire with a value of 1 plus the amount of new sessions between the two state events.

Arguments

- **state_from** (*object*) – The state where session counting should start.
- **state_to** (*object*) – The state where session counting should end.
- **state_*** (*object*) – The name and action of the state, e.g.

```
{
    state: 'states:foo',
    action: 'exit'
}
```

- **state_*.state** (*string*) – The name of the state. Required.
- **state_*.action** (*string*) – The state action. Can be one of `enter`, `exit`, `input`, `resume`, and `show`. Defaults to `enter`.
- **label** (*string*) – The label for the metric. Defaults to `sessions_between_$ACTIONFROM_$STATEFROM_$ACTIONTO_$STATETO` where `$ACTION_*` is the specified action, and `$STATE_*` is the name of the state with all characters that are not `a-zA-Z._` replaced with _.

`MetricsHelper.add.`**`sessions_until_state`**(*state*, *label*)
Adds an average metric that counts the amount of sessions taken to reach the specified state action.

Arguments

- **state** (*object*) – The name and action of the state to count actions for, e.g.

```
{
    state: 'states:foo',
    action: 'exit'
}
```

- **state.state** (*string*) – The name of the state to count the action for. Required.
- **state.action** (*string*) – The state action that should increment the count. Can be one of `enter`, `exit`, `input`, `resume`, and `show`. Defaults to `enter`.
- **label** (*string*) – The label for the metric. Defaults to `sessions_until_$ACTION_$STATE` where `$ACTION` is the specified action, and `$STATE` is the name of the state with all characters that are not `a-zA-Z._` replaced with _.

`MetricsHelper.add.`**`time_between_states`**(*state_from*, *state_to*, *label*)

Adds an average metric that measures the amount of time taken between two state events. Metric stored as milliseconds.

Arguments

- **state_from** (*object*) – The state where timing should start.
- **state_to** (*object*) – The state where timing should end.
- **state_*** (*object*) – The name and action of the state, e.g.

```
{
    state: 'states:foo',
    action: 'exit'
}
```

- **state_*.state** (*string*) – The name of the state. Required.
- **state_*.action** (*string*) – The state action. Can be one of `enter`, `exit`, `input`, `resume`, and `show`. Defaults to `enter`.
- **label** (*string*) – The label for the metric. Defaults to `time_between_$ACTIONFROM_$STATEFROM_$ACTIONTO_$STATETO` where `$ACTION_*` is the specified action, and `$STATE_*` is the name of the state with all characters that are not `a-zA-Z._` replaced with _.

`MetricsHelper.add.`**`total_sessions`**(*label*)

Adds inc and sum metrics that fires every time a new session is started.

Arguments

- **label** (*string*) – The label for the metric. Defaults to `total_sessions`

`MetricsHelper.add.`**`total_state_actions`**(*state*, *label*)

Adds inc and sum metrics that fires every time the specified state action is triggered.

Arguments

- **state** (*object*) – The name and action of the state to count actions for, e.g.

```
{
    state: 'states:foo',
    action: 'exit'
}
```

- **state.state** (*string*) – The name of the state to count the action for. Required.
- **state.action** (*string*) – The state action that should increment the count. Can be one of `enter`, `exit`, `input`, `resume`, and `show`. Defaults to `enter`.

- **label** (*string*) – The label for the metric. Defaults to `total_action_$ACTION_$STATE` where `$ACTION` is the specified action, and `$STATE` is the name of the state with all characters that are not `a-zA-Z._` replaced with `_`.

`MetricsHelper.add.`**`total_unique_users`**(*label*)

Adds inc and sum metrics that fires every time a new session is started with a new user.

Arguments

- **label** (*string*) – The label for the metric. Defaults to `unique_users`

`MetricsHelper.add.`**`tracker`**(*start_trigger*, *end_trigger*, *metrics*)

Allows the addition of multiple start/end metrics with one function.

Arguments

- **start_trigger** (*object*) – The action that should trigger the start of the metric.
- **end_trigger** (*object*) – The action that should trigger the end of the metric.
- ***_trigger** (*object*) – The name and action of the state, e.g.

```
{
    state: 'states:foo',
    action: 'exit'
}
```

- **metrics** (*object*) – The metrics and their labels that should be attached to the given triggers. Of the format `metric_type: metric_label`. `metric_type` can be one of `time_between_states`, `sessions_between_states`. e.g.

```
{
    time_between_states: 'time_between_a_and_b'
}
```

`MetricsHelper.add.`**`trigger`**(*trigger*, *metrics*)

Allows the addition of multiple metrics on one trigger event.

Arguments

- **trigger** (*object*) – The name and action of the state to trigger on, e.g.

```
{
    state: 'states:foo',
    action: 'exit'
}
```

- **metrics** (*object*) – The metrics and their labels that should be attached to the given trigger. Of the format `metric_type: metric_label`. `metric_type` can be one of `sessions_until_state`, `total_state_actions`. e.g.

```
{
    sessions_until_state: 'sessions_until_foo',
    total_state_actions: 'total_bar_completions'
}
```

CHAPTER 2

Configuring the dashboard

For the full documentation on dashboards within Vumi Go, please refer to the Vumi Go Dashboard documentation. This section of the documentation will describe how to use each of the metrics helper functions in the context of creating a Vumi Go dashboard.

2.1 Introduction

For all of the following sections, it is assumed that the metrics are being applied to this sample Vumi Go JavaScript Sandbox application:

```
var vumigo = require('vumigo_v02');
var MetricsHelper = require('go-jsbox-metrics-helper');

var App = vumigo.App;
var Choice = vumigo.states.Choice;
var ChoiceState = vumigo.states.ChoiceState;
var EndState = vumigo.states.EndState;

var SimpleApp = App.extend(function(self) {
    App.call(self, 'states:start');

    self.init = function() {
        new MetricsHelper(self.im);
        // Add metrics here
    };

    self.states.add('states:start', function(name) {
        return new ChoiceState(name, {
            question: 'Tea or coffee?',

            choices: [
                new Choice('tea', 'Tea'),
                new Choice('coffee', 'Coffee')],

            next: function(choice) {
                return {
                    tea: 'states:tea',
                    coffee: 'states:coffee'
                }[choice.value];
            }
        });
```

```
    });

    self.states.add('states:tea', function(name) {
        return new EndState(name, {
            text: 'Meh. Bye.',
            next: 'states:start'
        });
    });

    self.states.add('states:coffee', function(name) {
        return new EndState(name, {
            text: 'Cool :) Bye.',
            next: 'states:start'
        });
    });
});
```

2.2 Total Unique Users and Sessions

For the functions `total_unique_users()` and func:*total_sessions*, the following is an example of using the functions to add metrics to the basic application:

```
new MetricsHelper(self.im)
    .add.total_unique_users('unique_users')
    .add.total_sessions('total_sessions');
```

This will add four new metrics; *unique_users*, a metric with a *last* aggregation method that contains the current sum of unique users, and *unique_users.transient*, a metric with the *sum* aggregation method that is fired every time a new unique user accesses the service, *total_sessions* a metric with the *last* aggregation method which will contain the total amount of sessions, and *total_sessions.transient*, which is a metric with the *sum* aggregation method which fires every time a new session is started.

The following is an example for use of these metrics in the Vumi Go Dashboard:

```
{
    "type": "diamondash.widgets.lvalue.LValueWidget",
    "name": "Total new unique users",
    "time_range": "1d",
    "target": {
        "metric_type": "account",
        "store": "teaorcoffee",
        "name": "unique_users",
        "aggregator": "last"
    }
},
{
    "type": "diamondash.widgets.graph.GraphWidget",
    "name": "Total sessions over the past 30 days",
    "width": 12,
    "time_range": "30d",
    "bucket_size": "1d",
    "metrics": [
        {
            "name": "Unique Users",
            "target": {
                "metric_type": "account",
```

```
                "store": "teaorcoffee",
                "name": "total_sessions.transient",
                "aggregator": "sum"
            }
        }
    ]
}
```

The first widget will produce a text block with the total unique users over all time, with a comparison to the value from one day ago.

The second widget will produce a line graph, showing the total new sessions per day for the last 30 days.

2.3 Total State Actions and Sessions Until State

The functions `total_state_actions()` and `sessions_until_state()` are best invoked using the `trigger()` function. The following is an example of using this function to add metrics to the basic application:

```
new MetricsHelper(self.im)
    .add.trigger({
        action: 'enter',
        state: 'states:tea'
    }, {
        total_state_actions: 'total_tea',
        sessions_until_state: 'sessions_per_tea'
    })
```

This will add three new metrics; *total_tea*, a metric with a *last* aggregation method that contains the total amount of *enter* events on the *states:tea* state; *total_tea.transient*, a metric with a *sum* aggregation method that is fired every time the *enter* event on the *states:tea* state is triggered; and *sessions_per_tea*, a metric with an *avg* aggregation method, which is triggered every time the *enter* event of the *states:tea* state is triggered, and contains the number of sessions taken to get to that event.

The following is an example of using these metrics in a Vumi Go Dashboard:

```
{
    "type": "diamondash.widgets.lvalue.LValueWidget",
    "time_range": "1d",
    "name": "Total tea drinkers",
    "target": {
        "metric_type": "account",
        "store": "teaorcoffee",
        "name": "total_tea",
        "aggregator": "last"
    }
},
{
    "type": "diamondash.widgets.lvalue.LValueWidget",
    "time_range": "1d",
    "name": "Average sessions until tea is chosen",
    "target": {
        "metric_type": "account",
        "store": "teaorcoffee",
        "name": "sessions_per_tea",
        "aggregator": "avg"
    }
},
```

```
{
    "type": "diamondash.widgets.graph.GraphWidget",
    "name": "Weekly tea consumption for the last 30 days",
    "width": 12,
    "time_range": "30d",
    "bucket_size": "7d",
    "metrics": [{
        "name": "Tea",
        "target": {
            "metric_type": "account",
            "store": "teaorcoffee",
            "name": "total_tea.transient",
            "aggregator": "sum"
        }
    }]
}
```

The first widget will create a text block showing the total amount of tea drinkers, with a comparison to the total amount of tea drinkers from one day ago.

The second widget will show the average amount of sessions taken to get to the tea state, with a comparison to the amount from one day ago.

The last widget is a line graph that shows the amount of tea drinkers, grouped by week, over the last 30 days.

2.4 Time Between States

The function `time_between_states()` is best invoked using the `tracker()` function. The following is an example application within the basic application:

```
new MetricsHelper(self.im)
    .add.tracker({
        action: 'enter',
        state: 'states:start'
    }, {
        action: 'enter',
        state: 'states:coffee'
    }, {
        time_between_states: 'time_between_start_and_coffee'
    })
```

This will add one metric, *time_between_start_and_coffee*, with the aggregation method *avg*, which stores the average time in milliseconds taken to get from the *enter* event of the *states:start* state to the *enter* event of the *states:coffee* state.

The following is an example of using this metric within the Vumi Go Dashboard:

```
{
    "type": "diamondash.widgets.graph.GraphWidget",
    "name": "Average time taken to choose coffee",
    "time_range": "30d",
    "bucket_size": "7d",
    "metrics": [{
        "name": "Coffee",
        "target": {
            "metric_type": "account",
            "store": "teaorcoffee",
```

```
            "name": "time_between_start_and_coffee",
            "aggregator": "avg"
        }
    }]
}
```

This will create a graph widget which shows the average time taken to choose coffee per week, for the last 30 days.

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

M