StackIt Documentation

Release 0.1

Guillaume Luchet

May 01, 2013

CONTENTS

Push data into stacks for later processing.

CHAPTER

ONE

GETTING STARTED

1.1 Loading the library

Register the library using the StackitAutoloader object:

```
require_once 'library/StackIt/Autoloader.php';
StackIt\Autoloader::register();
```

1.2 Configure your stacks

Create an ini config file to setup your first stack:

```
;; StackIt stack config for the stack "my-stack"
[my-stack]

;; The key value store mapper to use
kvs="StackIt\Kvs\Redis"

;; The processor implementation to use to process the stack
processor="StackIt\Processor\PostgreSQL\CopyFromStdin"

;; Minimum interval between 2 iterations of the stack processor
;; (seconds) (0 to not define minimum interval, default 0)
interval=30

;; Max number of execution of the stack processor in the same daemon
;; instance (0 if no limit, default 0)
max_execution=2
```

Load your config:

```
StackIt\Stack::setConfig('/path/to/config.ini');
```

You can replace the ini config file using a Php array:

```
$config = array(
    'my-stack' => array(
        'kvs' => 'StackIt\Kvs\Redis',
        'processor' => 'StackIt\Processor\PostgreSQL\CopyFromStdin',
        'interval' => 30,
        'max_execution' => 2,
    ),
```

```
);
StackIt\Stack::setConfig($config);
```

1.3 Put in stack

Now StackIt know your stacks, you can push data to it:

```
StackIt\Stack::push('my-stack', array(
    'customer_id' => 123,
    'order_amount' => 59.99,
    'order_date' => date('Y-m-d H:i:s')
));
```

1.4 Consume stacks

Then in a cron job, for example, run the daemon to process the stacks:

```
StackIt\Daemon::setConfig('/path/to/config.ini');
$d = StackIt\Daemon::singleton();
$d->run();
```

CHAPTER

TWO

CONFIGURATION

Each stack must be defined in the stacks config.

- **kvs** The key value store to use to store the data
- **processor** The processor to use to consume the data
- **interval** Minimum interval in seconds between two iterations of the stack processor
- **max_execution** Maximum number of execution of the processor in the same daemon instance

In addition each stack KVSs and processors come with their own config entries. Those entries are detailled in the relevant sections.

CHAPTER

THREE

DATA PROVIDERS

3.1 Built-in data providers

A data provider is an object used to push data into a stack.

3.1.1 Redis

The StackItKvsRedis data provider use the phpredis extension. This data provider require the following config entries

- **redis_host** The Redis server host
- **redis_port** The Redis server port

Example of configuration:

```
;; The Redis server host
redis_host="127.0.0.1"

;; The Redis server port
redis_port=6379
```

Dependances:

- A Redis server
- The phpredis extension

3.1.2 PRedis

If you can not install the phpredis extension you can use the StackItKvsPRedis data processor which use the predis client. The required config entries are

- **redis_host** The Redis server host
- **redis_port** The Redis server port

Dependances:

- A Redis server
- The predis client

3.2 Create your own data provider

This section is not done

CHAPTER

FOUR

DATA CONSUMERS

4.1 Build-in data consumers

4.1.1 Console Exec

Call exec for each item in the stack the log the output using the defined method.

Config options:

- **exec_log_method** The method to use to log the exec output, available values are
  - *syslog* Log in syslog
  - *file* Log in file
  - *none* do not log
- **exec_log_file** The path to the log file to use if log method is *file*

4.1.2 MySQL Insert

The StackItProcessorMysqlInsert provider perform an insert for each data in the stack

Dependances:

- MySQL database

Config options:

- **mysql_server** The MySQL server host
- **mysql_username** The MySQL username
- **mysql_password** The MySQL password
- **mysql_database** The database to use
- **mysql_insert_table** The table where insert the data
- **mysql_insert_columns** A coma separated list of columns name

Sample config:

```
mysql_server=locahost
mysql_username=foo
mysql_password=bar
mysql_database=mydb
```

```
mysql_insert_table=order_log
mysql_insert_columns=customer_id,order_amount,order_date
```

4.1.3 MySQL MassInsert

The StackItProcessorMySQLMassInsert provider will use the same config entries than the StackItProcessorMySQLInsert but all the stacked data will be inserted in one query.

4.1.4 PostgreSQL Insert

The StackItProcessorPostgreSQLInsert processor perform an insert query for each data in the stack

Dependances:

- PostgreSQL database

Config options:

- **pg_conn_str** The PostgreSQL connection string
- **pg_insert_table** The table where insert the stacked data
- **pg_insert_columns** The table columns

Config sample:

```
pg_conn_str="dbname=dbtest"
pg_insert_table="order_log"
pg_insert_columns="customer_id,order_amount,order_date"
```

4.1.5 PostgreSQL copyFromStdin

The StackItProcessorPostgreSQLCopyFromStdin will store the stacked data in a PostgreSQL database using a "copy from stdin" method rather than many inserts.

Config options:

- **pg_conn_str** The PostgreSQL connection string
- **pg_insert_table** The table where insert the stacked data
- **pg_insert_columns** The table columns

Config sample:

```
pg_conn_str="dbname=dbtest"
pg_insert_table="order_log"
pg_insert_columns="customer_id,order_amount,order_date"
```