Usage Manual for the backend bechmarktool web application

June 5, 2015

Summary

1 Introduction

This manual is a quick guide that explains how to manipulate data with the benchmarktool web application. It explains how to check the existing registers on the database, and how to add new benchmark sets and benchmark parameters.

2 Listing the existent registers on a table

To list all the registers on an existing table of the system, you must perform perform a GET on the following URL:
`https://<host_name>/<applicationname>/default/api/<table_name>.json`

In the current deployment scenario:
`https://gwt.eit.uni-kl.de/benchmarktool/default/api/<table_name>.json`

3 Adding new register to tables

3.1 Adding an option price

A register in the option price table is composed by an ID and a name. Since the ID fields are auto-incremented by default in our application, only the name should be defined. The type of the field "name" is string.

To add a new option price it is necessary to insert a tuple in the database. In our architecture, the way of doing this is performing a POST via HTTPS, since we are using a REST architecture. The data to be added should be in the payload field and then a POST request has to be sent to the URL:
`https://gwt.eit.uni-kl.de/benchmarktool/default/api/option_price.json`

In case of a new installation, the URL format should be:
`https://<host_name>/<applicationname>/default/api/option_price.json`

An example using the requests library for Python:

```
payload_option_price={"name":"OPTION_TYPE_CALL_DIGITAL"}
requests.post("https://gwt.eit.uni-kl.de/benchmarktool/default/
    api/option_price.json", data=payload_option_price)
```

3.2 Adding a barrier type

A register in the barrier table is composed by an ID and a name. Since the ID fields are auto-incremented by default in our application, only the name should be defined. The type of the field "name" is string. To add a new barrier type it is necessary to insert a tuple in the database. In our architecture, the way of doing this is performing a POST via HTTPS, since we are using a REST architecture. The data to be added should be in the payload field and then a POST request has to be sent to the URL:
`https://gwt.eit.uni-kl.de/benchmarktool/default/api/barrier.json`

In case of a new installation, the URL format should be:
`https://<host_name>/<applicationname>/default/api/barrier.json`

An example using the requests library for Python:

```
payload_barrier={"name":"BARRIER_TYPE_KNOCK_OUT"}
requests.post("https://gwt.eit.uni-kl.de/benchmarktool/default
    /api/barrier.json", data=payload_barrier)
```

3.3 Adding a market parameter

A market parameter is a tuple composed by the parameters that define a specific market. It is composed by the following parameters:

- correlation
- long run variance
- speed of revertion

- volatility of volatility
- spot price
- spot volatility
- riskless interest rate

To perform a simulation you must define all those parameters, since they are a fundamental part of the market simulation algorithms. The way to define them is to add a tuple in the database by performing a POST via HTTPS, since we are using a REST architecture. The data to be added should be in the payload field and then a POST request has to be sent to the URL:
`https://gwt.eit.uni-kl.de/benchmarktool/default/api/market_parameters.json`

In case of a new installation, the URL format should be:
`https://<host_name>/<applicationname>/default/api/market_parameters.json`

An example using the requests library for Python:

```
payload_market_param={"spot_price": 100.0, "long_run_variance": 0.09, "volatility_of_volatility": 1.0, "spot_volatility": 0.09, "riskless_interest_rate": 0.05, "correlation": -0.3, "speed_of_revertion": 2.0}
requests.post("https://gwt.eit.uni-kl.de/benchmarktool/default/api/market_parameters.json", data=payload_market_param)
```

The respective types of the parameters are:

- correlation: double
- long_run_variance: double
- speed_of_revertion: double
- volatility_of_volatility: double
- spot_price: double
- spot_volatility: double
- riskless_interest_rate: double

3.4 Adding an option parameter

An option parameter is a tuple composed by the parameters that define an specific option. It is composed by the following parameters:

- option type
- strike price

- time to maturity
- lower barrier type
- lower barrier value
- upper barrier type
- upper barrier value

To perform a simulation you must define all those parameters, since they are a fundamental part of the benchmarks. The way to define them is to add a tuple in the database by performing a POST via HTTPS, since we are using a REST architecture. The data to be added should be in the payload field and then a POST request has to be sent to the URL:
`https://gwt.eit.uni-kl.de/benchmarktool/default/api/option_parameters.json`

In case of a new installation, the URL format should be:
`https://<host_name>/<applicationname>/default/api/option_parameters.json`

An example using the requests library for Python:

```
payload_option_param = {"time_to_maturity": 1.0, "strike_price":
    100.0, "lbarrier_value": 90.0, "ubarrier_value": 110.0, "
    option_type": 2, "lbarrier_type": 1, "ubarrier_type": 1}
requests.post("https://gwt.eit.uni-kl.de/benchmarktool/default/
    api/option_parameters.json", data=payload_option_param)
```

The respective types of the parameters are:

- option_type: integer (FK to table option_price)
- strike_price: double
- time_to_maturity: double
- lbarrier_type: integer (FK to table barrier)
- lbarrier_value: double
- ubarrier_type: integer (FK to table barrier)
- ubarrier_value: double

3.5 Benchmark Set

A benchmark set is composed by name, description, market parameters and option parameters. As it is an REST architecture, to add a new benchmark set it is necessary to perform a POST via HTTPS. The data to be added should be in the payload field and then a POST request has to be sent to the URL:

`https://gwt.eit.uni-kl.de/benchmarktool/default/api/benchmark_set.json`

In case of a new installation, the URL format should be:

`https://<host_name>/<applicationname>/default/api/benchmark_set.json`

An example using the requests library for Python:

```
payload_benchmark_set = {"mkt_parameters": 1, "description": "
    Benchmark set detailed description.", "opt_parameters": 1, "
    name": "Benchmark 1"}
requests.post("https://gwt.eit.uni-kl.de/benchmarktool/default/
    api/benchmark_set.json", data=payload_benchmark_set)
```

The respective types of the parameters are:

- name: string
- description: long string
- mkt_parameters: (FK to table option_parameters)
- opt_parameters: (FK to table market_parameters)

4 Starting a Working Node

Working nodes are automatically registered into the system once the web2py process is started. To be able to disconnect from the remote connection, it is advisable to first start a screen and then start web2py inside it. The words inside the curly brackets must be replaced with the appropriated ones.

```
[web2py: ~]# screen
[web2py: ~]# python {full path to web2py}/web2py.py -a '<recycle
    >' -i {interface's IP} -p {port}
web2py Web Framework
Created by Massimo Di Pierro, Copyright 2007-2015
Version 2.9.12-stable+timestamp.2015.01.17.06.11.03
Database drivers available: sqlite3, pymysql, pg8000, imaplib

please visit:
        http://127.0.0.1:8000/
use "kill -SIGTERM 19414" to shutdown the web2py server

starting scheduler for "benchmarktool"...

Currently running 1 scheduler processes
Processes started
```