

E.M.T.E

(Exchange Matching and Trading Engine)

Requirement

- Install [web2py](#).
 - Install [tornado web server](#). (Tornado v4.3)
 - Web browsers.
-

How to run EMTE-TRADING application.

1. *Start web2py.* (For more information, visit [web2py overview](#)) * Download, uncompress web2py, and run web2py server with your admin password `python web2py.py` * Install EMTE-TRADING application.

For simplicity, we assume that the web2py server is run at <http://127.0.0.1:8000>. Open web browser, and go to <http://127.0.0.1:8000/admin/default/site>. Under Upload and install packed application, name this application and enter the git link. Click install and your EMTE-TRADING applicaiton is ready to run.

2. *Navigate to application module, and run matching.py.* Matching.py takes 2 arguments: * **-p**: port number it listen to. Each ticker has different port number. * **-t**: a ticket symbol.

Users should insert valid value to the product table before calling **matching.py**. The example below assumed that all tickers symbol is already stored in the database.

```
cd applications/emte/modules/
```

Each ticker is associated with different port number.

```
python matchingserver.py -p 8888 -t intc python matchingserver.py -p 8880 -t gs
```

3. *Start log2db daemon (one per product)*

Rationale, decouples the job of logging to db from the job of logging to file.

```
python web2py.py -S emte -M -R applications/emte/modules/log2db.py -A intc
```

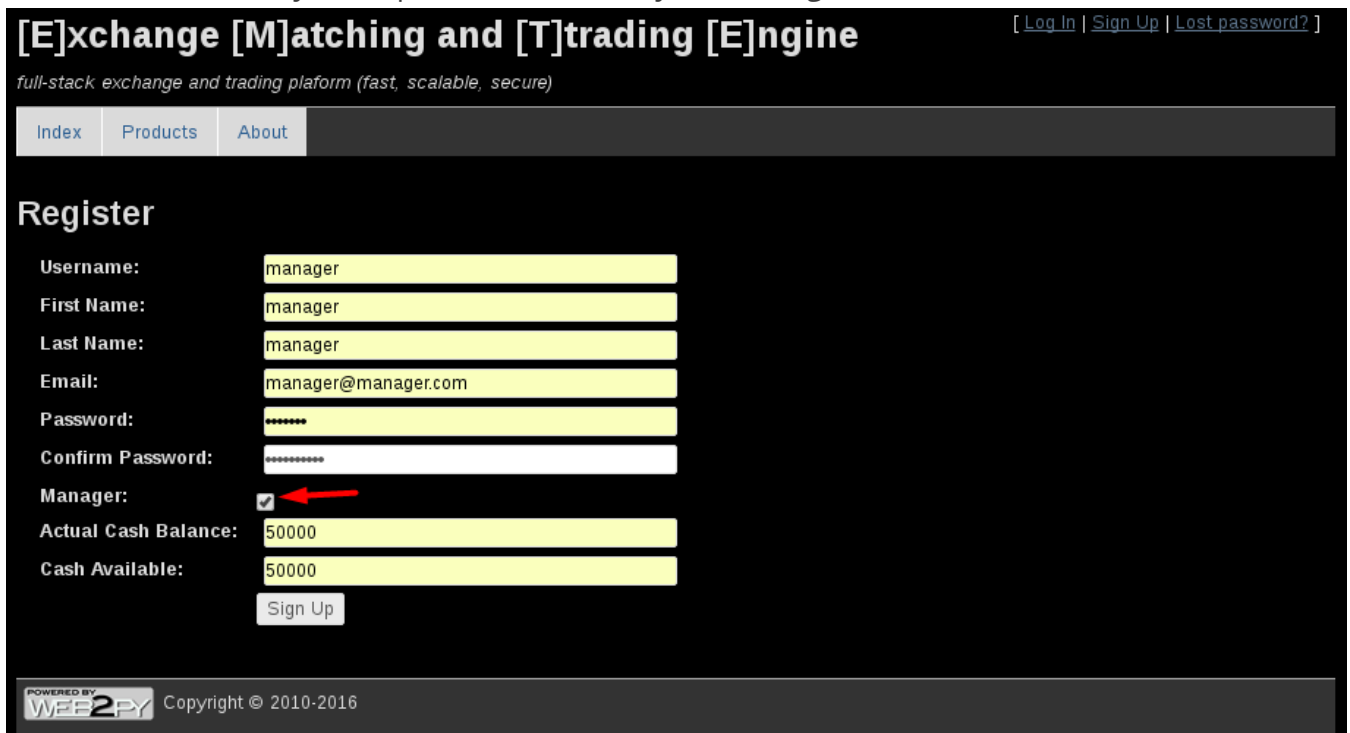
Overview

EMTE is a Exchange Matching Engine (think of NYSE and Nasdaq) and a Trading Platform that connects to it. It is built in Python with web2py and Tornado. It uses comet via html5 websockets for asynchronous information delivery. The web interface uses jQuery and processing.js for drawing.

1. The Exchange Matching Engine

As the name implied, EMTE function can be divided into two major roles, the exchange matching engine, and the trading platform. In order to start the exchange matching engine, users run web2py EMTE application, open the application in your favorite web browser.

- **First time users:** When you run the application for the first time, don't run **matching.py** just yet. Make sure that you follow these steps:
 - Signup. There are 2 modes for users when they sign up: *manager*, and *clients/traders*. As a *manager*, you have permission to create/register any product to trade. While the *clients/traders* only trade products created by the *manager*.



The screenshot shows the web interface of the EMTE application. At the top, the title is "[E]xchange [M]atching and [T]rading [E]ngine" with links for [Log In | Sign Up | Lost password?]. Below the title is a navigation bar with "Index", "Products", and "About" tabs. The main content area is titled "Register" and contains a form with the following fields: Username (manager), First Name (manager), Last Name (manager), Email (manager@manager.com), Password (masked with dots), Confirm Password (masked with dots), Manager (checked with a red arrow pointing to it), Actual Cash Balance (50000), and Cash Available (50000). A "Sign Up" button is at the bottom of the form. The footer shows "POWERED BY WEB2PY" and "Copyright © 2010-2016".

- Register product (log in as manager). Make sure that you match each port of registered ticker with the port when run **matching.py** here. Note that each port is assigned with different port number. **Matching.py** uses websocket protocol to handle request from client. Therefore, you need to specify **web socket protocol** in **Ws Url** form. Simply click on *Products* tab. The form to insert product is shown as below:

Register new product

remember to start applications/modules/machingengine.py <name>

Name:

Description:

Unit Price:

Post Url:

Quote Url:

Ws Url:

Register new product

remember to start applications/modules/machingengine.py <name>

Name:

Description:

Unit Price:

Post Url:

Quote Url:

Ws Url:

- Trading matching engine web interface. Before open the web interface to trade, remember to run `matching.py`. The web interface communicates with `matching.py` base on Tornado web socket. It acts as another server communicate with clients. Therefore, the matching engine must start to listen to clients.

[E]xchange [M]atching and [T]rading [E]ngine

Welcome trader1 [[Log Out](#) | [Profile](#) | [Password](#)]

full-stack exchange and trading plaform (fast, scalable, secure)

[Index](#)
[Products](#)
[About](#)

Tradable Products

appl	[trade] [profits/loss]	Apple...
gs	[trade] [profits/loss]	Goldman Sachs Group Inc...

POWERED BY **WEE2PY** Copyright © 2010-2016

When entering the *Products* page, this is the response from matching engine.

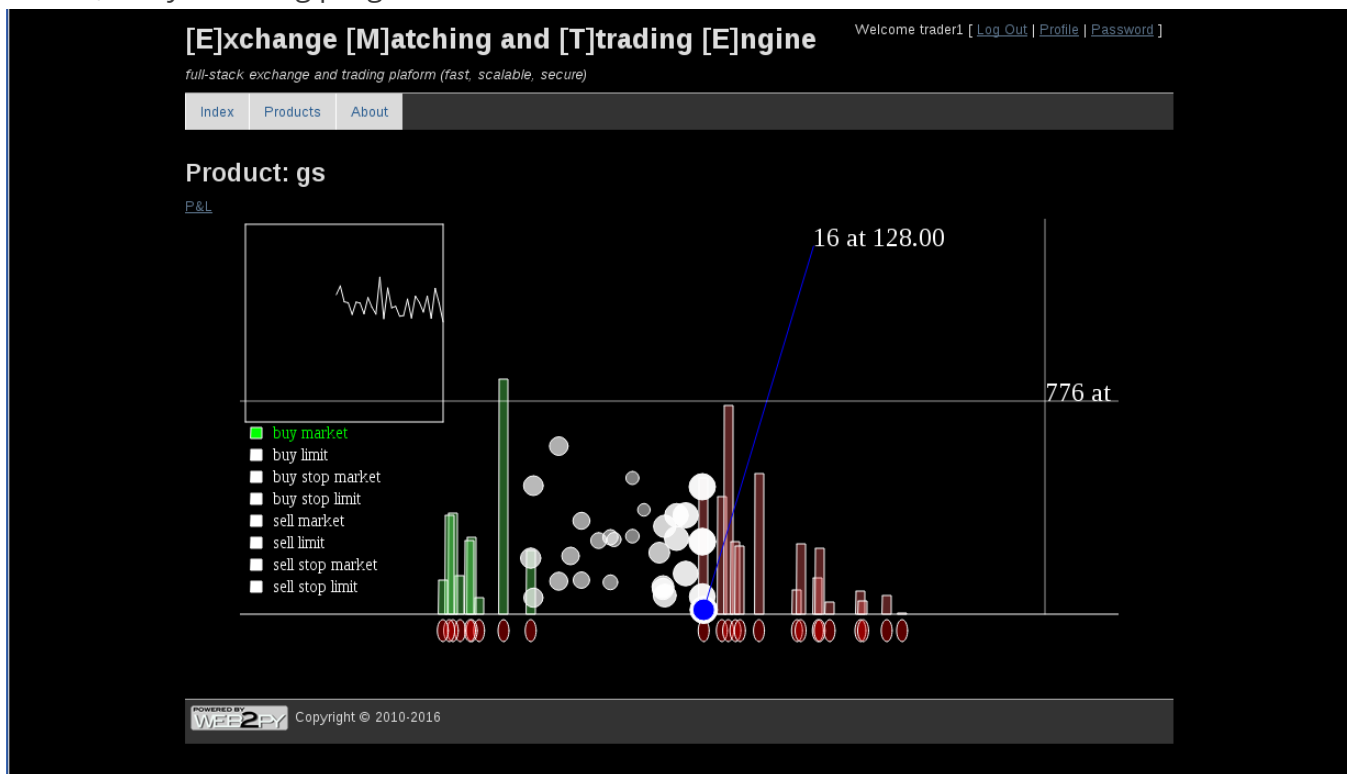
```
htran:Devops$ cd web2py/applications/emte/modules/
htran:modules$ python matchingserver.py -p 8881 -t gs
client connected via websocket
client disconnected
```

EMTE allows users to

trade when there is record of product in the database.

User can submit order by check in the order options box, then click on the screen. When you hover the mouse on the page, there is a crosshair that show you the quantity at specific price. The horizontal axis tracks the price of our financial product. Price increases as the crosshair hovering to

the right, and decreases as the crosshair hovering to the left. Similarly, if users move the crosshair up, the quantity increase, and decrease otherwise. Users can submit order by clicking on the screen, or by a trading program.



2. The Trading Platform

As mention above, users can manually clicking on the screen at the price they are willing to pay. One advance feature of EMTE is that it allows *algorithmic, and automated trading*.

Users choose to utilize such feature can implement their trading algo and communicate with the matching engine. The example below is the random trade of Apple Inc. stock from robot trader. Note that *port 8888* is currently associated with *app/ticker* in our database.

Traders can track their account balance any time, simply by clicking to the **P&L**. This open a page that updates balance account in real time.

```
htran:modules$ python robot_trader.py -h
Usage: robot_trader -p 8888 -o 0 -k <hmac_key>

Options:
  -h, --help                show this help message and exit
  -p PORT, --port=PORT      socket
  -o OWNER, --owner=OWNER   the user id of the robot
  -k HMAC_KEY, --hmac_key=HMAC_KEY the hmac_key to sign orders
  -w WAIT_TIME, --wait_time=WAIT_TIME time between two trades in seconds

htran:modules$
htran:modules$ python robot_trader.py -p 8888 -w 2.0
order #20 from 0:buy 600@0 (0.005433seconds)
order #21 from 0:sell 906@0 (0.005307seconds)
```

[For more information](#)

3. Demo trading (Optional)

The **demo.py** in modules directory can be run to register manager, traders, stock, and submit order. To log in and view the simulation, the password is also the username.

Run demo.py in web2py directory

```
python web2py.py -S <app name> -M -R applications/<app name>/modules/demo.py
```

To make sure all matching server and robot trader running process is close, run stop_all.sh to close ALL matching servers, and robot traders.

```
bash stop_all.sh
```
