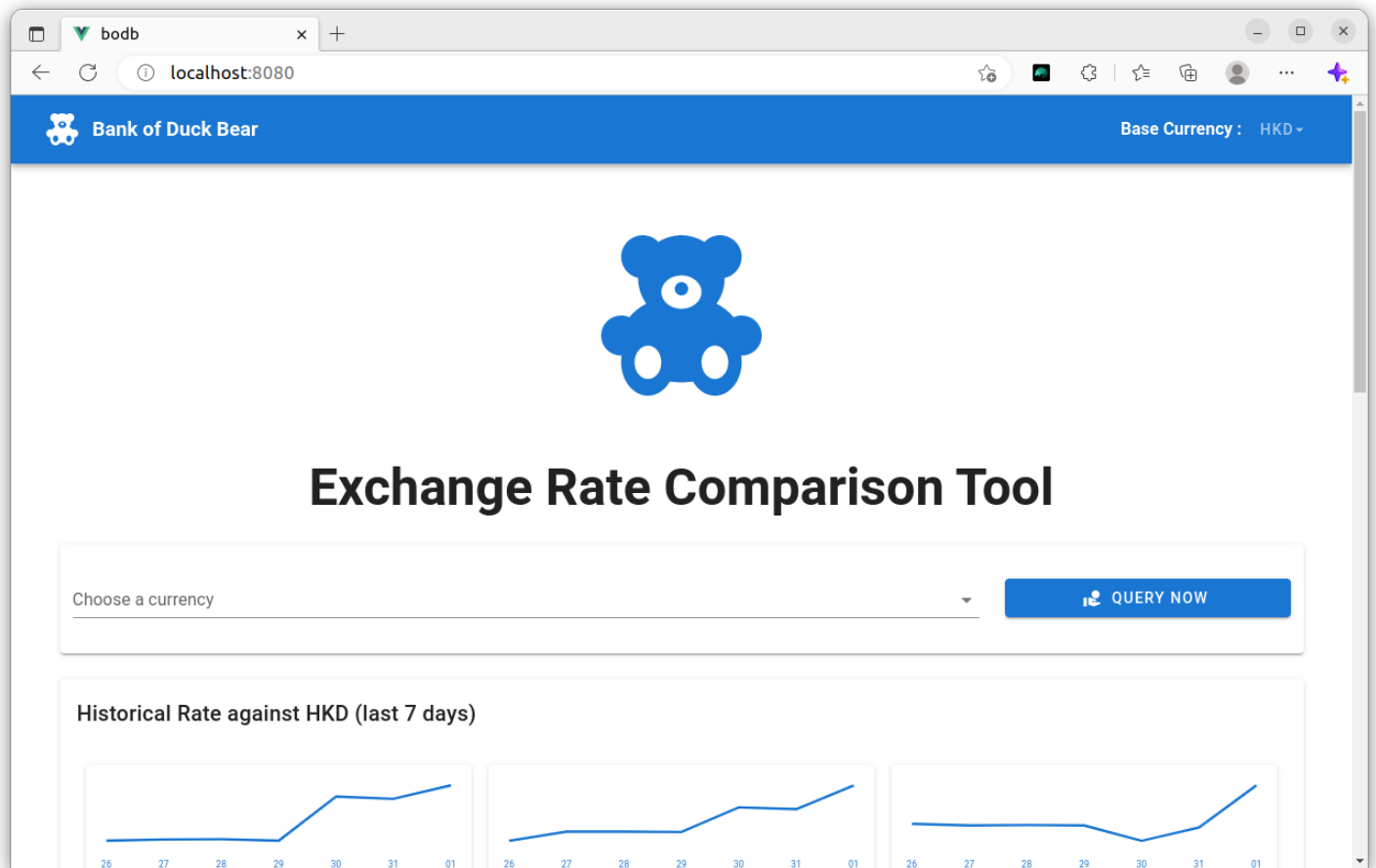


# BODB Exchange Rate Comparison Tool

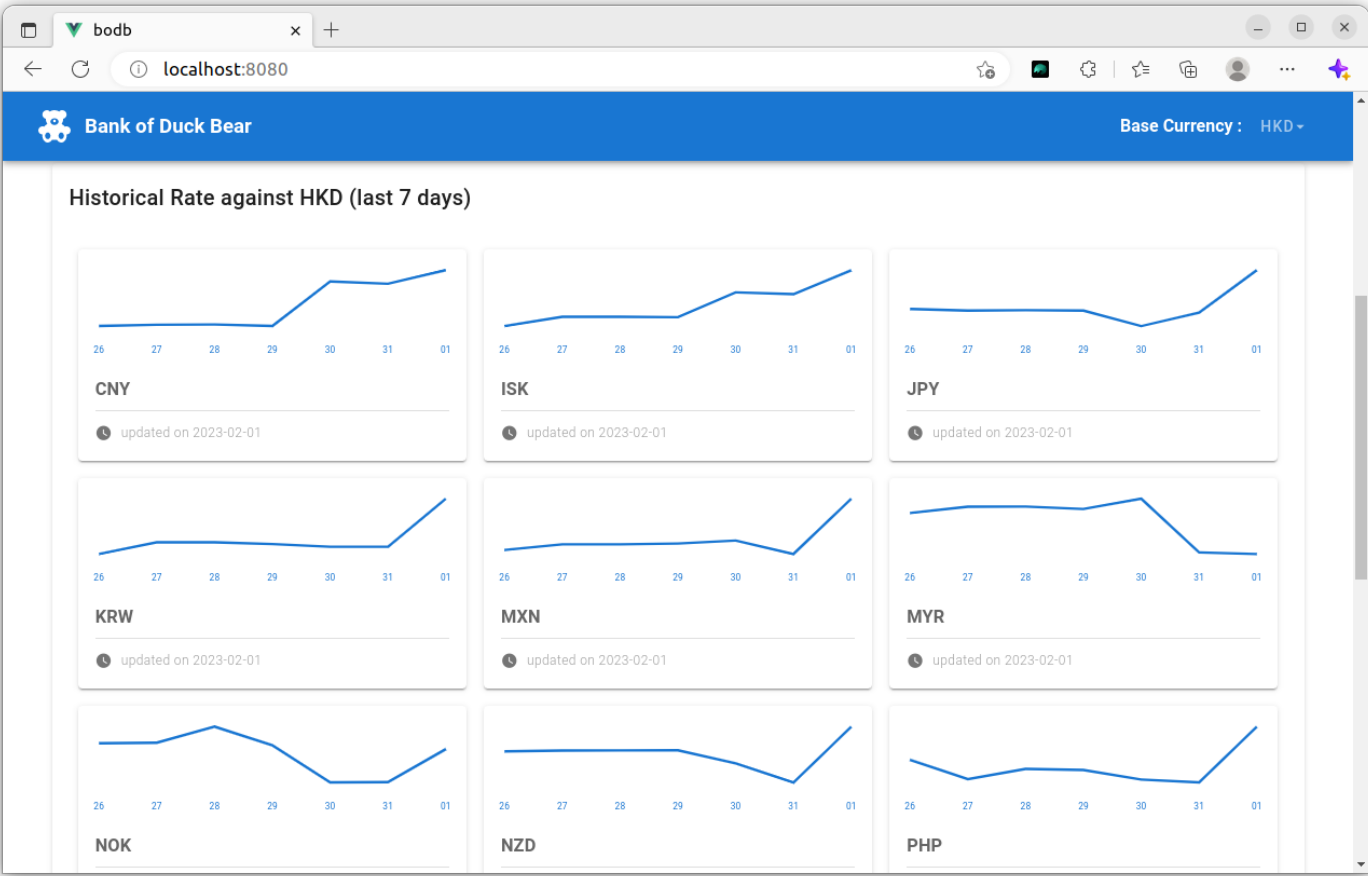
## Introduction



Bank of Duck Bear Exchange Rate Comparison Tool helps you know about the latest exchange rate trend and lists telegraphic transfer buy, telegraphic transfer sell, banknotes buy, banknotes sell provided by major banks, so that you can make a decision on where to buy your currencies!

The tool is currently deployed on <http://47.115.201.81:8080> for testing. Feel free to try it!

# Taking a Glance at Recent ER Trend



# Find a Bank with the Best Offer

The screenshot shows the 'Bank of Duck Bear' web application with the 'Currency' dropdown set to 'CNY'. The URL is 'localhost:8080/result?base\_ccy=HKD&ccy=CNY'. The page title is 'CNY'. On the right, there are definitions for 'TT Buy', 'TT Sell', 'Note Buy', and 'Note Sell'. The main content is a table with 7 columns: Bank, TT Buy, TT Sell, Note Buy, Note Sell, Last Updated, and Feeling data out-dated?. The table lists four banks: HSBC, HANG SENG BANK, Bank of Duck Bear, and Bank of China (Hong Kong). Each row has an 'UPDATE' button in the 'Feeling data out-dated?' column. At the bottom right, there is a 'Rows per page' dropdown set to '10' and a '1-4 of 4' indicator.

Bank	TT Buy	TT Sell	Note Buy	Note Sell ↓	Last Updated	Feeling data out-dated?
HSBC	1.1467	1.1620	1.1413	1.1708	2023-02-06 14:44:47	UPDATE
HANG SENG BANK	1.1469	1.1629	1.1369	1.1669	2023-02-06 14:41:00	UPDATE
Bank of Duck Bear	1.1652	1.1652	1.1652	1.1652	2023-02-03 03:21:25	
Bank of China (Hong Kong)	N/A	N/A	1.1670	1.1380	2023-02-06 22:47:57	UPDATE

# Request for Latest Data

bank of duck bear

USD

QUERY NOW

Base Currency : HKD

USD

TT Buy: Telegraphic Transfer Bank Buy

TT Sell: Telegraphic Transfer Bank Sell

Note Buy: Banknotes Bank Buy

Note Sell: Banknotes Bank Sell

Bank	TT Buy ↑	TT Sell	Note Buy	Note Sell	Last Updated	Feeling data out-dated?
Bank of China (Hong Kong)	N/A	N/A	7.8900	7.8100	2023-02-06 23:07:58	UPDATE
HSBC	7.8155	7.8772	7.7833	7.9100	2023-02-06 14:44:47	UPDATE
HANG SENG BANK	7.8310	7.8630	7.7700	7.8940	2023-02-06 15:01:00	UPDATE
Bank of Duck Bear	7.8447	7.8447	7.8447	7.8447	2023-02-03 03:21:25	

Rows per page:

10

1-4 of 4

<

>

Your update request received.

CLOSE

# Mobile Supported

11:14 PM

Bank of Duck Bear

HKD

Bank of Duck Bear

Exchange Rate Comparison Tool

Choose a currency

QUERY NOW

Historical ER against HKD (last 7 days)

Sort by

TT Buy

Bank

Bank of China (Hong Kong)

TT Buy

N/A

TT Sell

N/A

Note Buy

0.0601

Note Sell

0.0587

Last Updated

2023-02-06 23:12:59

Feeling data out-dated?

UPDATE

Bank

HSBC

TT Buy

0.0587

TT Sell

0.0598

Note Buy

0.0587

Note Sell

0.0604

Last Updated

2023-02-06 14:44:47

Feeling data out-dated?

UPDATE

Bank

Bank of Duck Bear

TT Buy

TT Sell

Note Buy

Note Sell

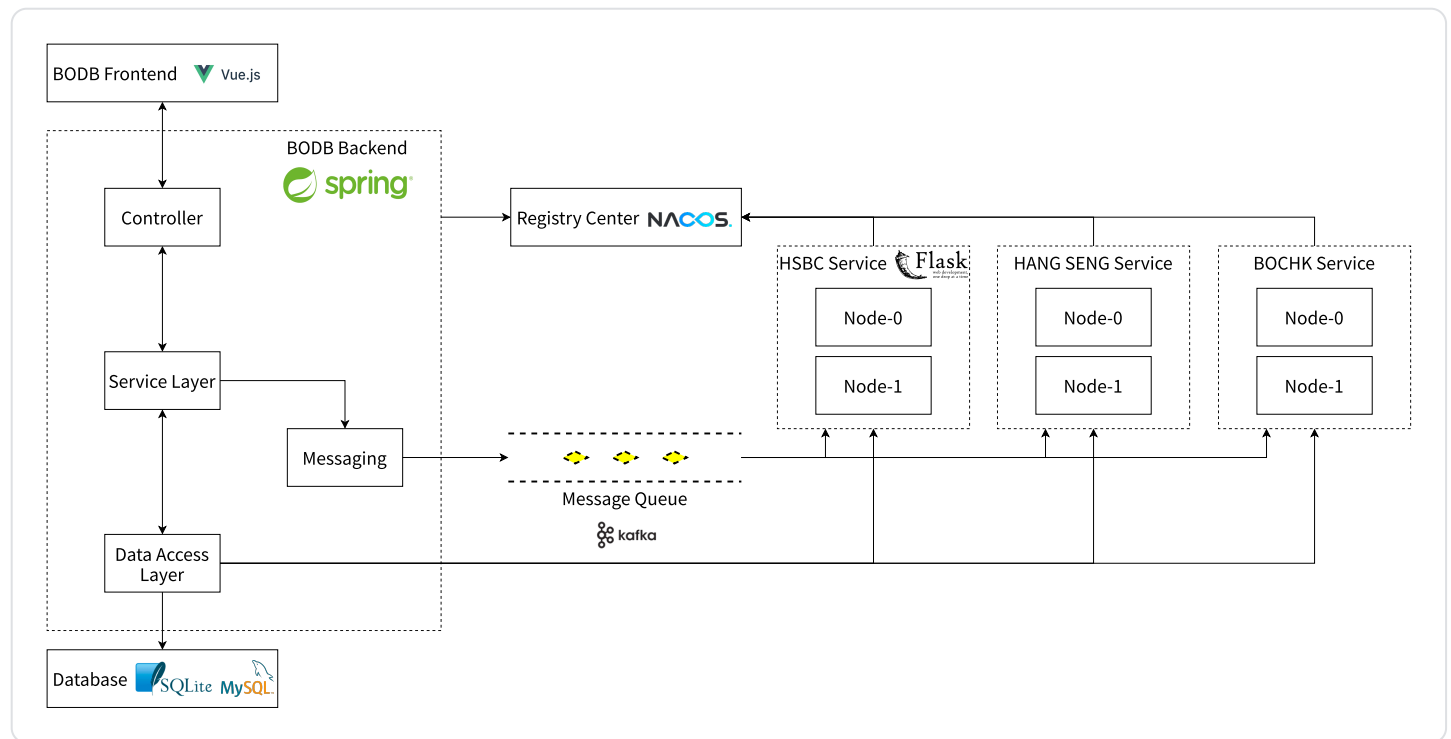
Last Updated

Feeling data out-dated?

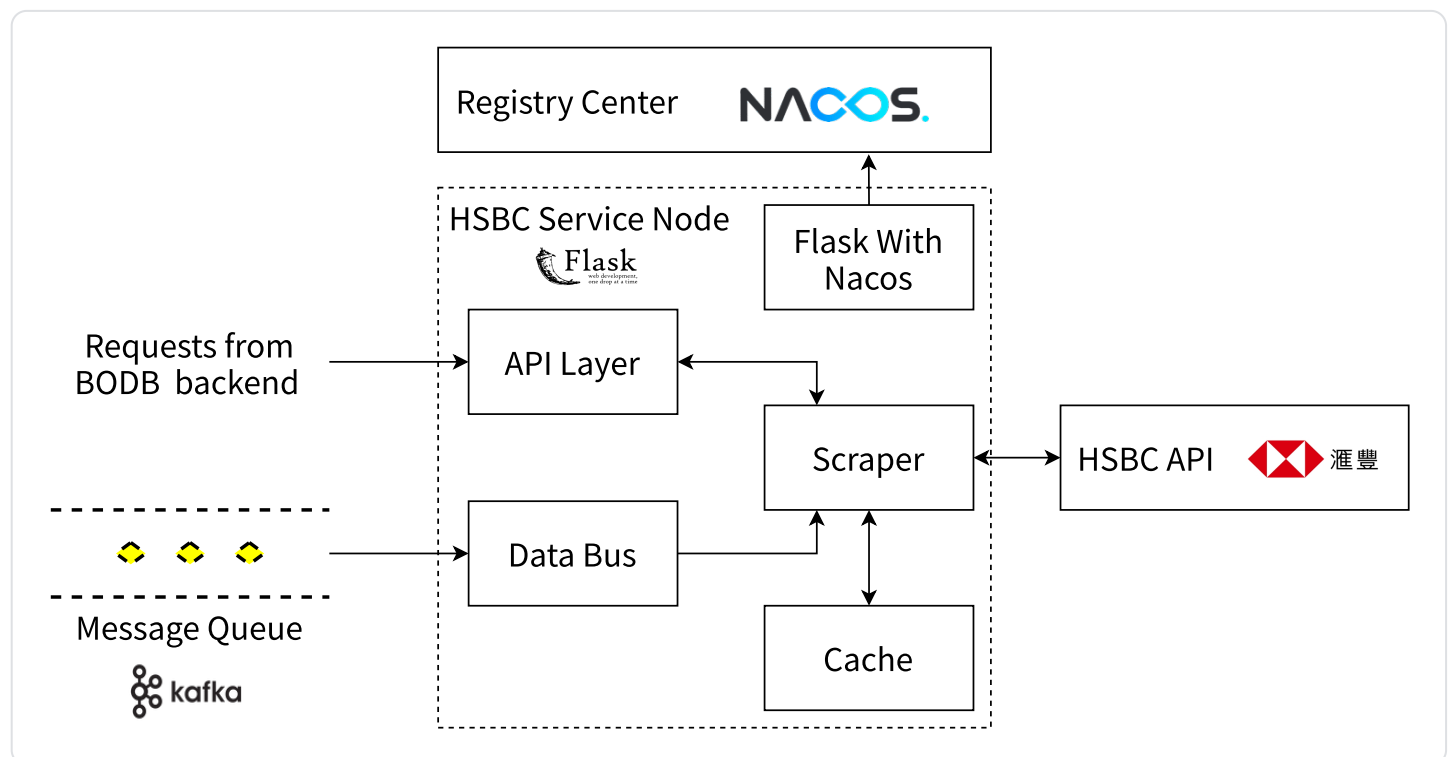
UPDATE

# Architecture

## Overall



## Inside a Third Party Bank Service



Among the overall architecture, BODB frontend is the user interface, developed by [Vue.js](#). It requests BODB backend based on [Spring](#). BODB backend, together with third party bank services (HSBC, BOCHK, HANG SENG), are automatically registered to [Nacos](#) on startup, so that BODB backend is allowed to send load-balanced REST requests to third party bank services.

Also, BODB maintains some self-owned data, such as our own exchange rates, stored in SQLite (instead of MySQL to reduce complexity when deploying the whole system). Meanwhile, when a user sends a request to update data, the backend produces a message to the message queue Kafka.

Inside a third party bank service instance, the API layer is designated for handling requests from BODB backend. The data bus consumes messages of updating requests from users. The scraper deals with scraping requests from both API Layer and Data Bus. It caches requested data for a short period of time, so that we do not need to turn to HSBC API frequently. After all, APIs used in this project are not public paid APIs.

Third party bank services are developed based on [Flask](#). However, Flask is merely a web framework that cannot be integrated with [Nacos](#) registry center. Worse still, although Nacos is one of the most popular registry centers in Java world, its Python SDK was not updated for a long time. On this occasion, I first fixed the bugs of Nacos Python SDK that I met, and then developed a package called *Flask with Nacos* so that I can combine Flask and Nacos together and let Flask register to Nacos automatically on startup.

## Deploy in Development Environment from Source Code

In case of any accident in deployment, the system is deployed on <http://47.115.201.81:8080> for testing.

### Requirements

1. JDK 1.8 and Maven 3.
  2. Python 3.7.
  3. NodeJS 18.14.0 and NPM.
- Some of the following frameworks also rely on Linux to work, but are allowed to be deployed in Windows by Windows Subsystem of Linux.

### Deploy [Nacos](#) (Require JDK 1.8+)

1. Download Nacos 1.3.2 (<https://github.com/alibaba/nacos/releases/download/1.3.2/nacos-server-1.3.2.zip>).
2. Unzip the downloaded file and

```
1 cd nacos
2 ./bin/startup.sh -m standalone      # for Linux
3 ./bin/startup.bat -m standalone    # for Windows
```

## Deploy **Kafka** (Require JDK 1.8+, must run on Linux)

1. Dowload Kafka with Scala 2.4.1 with Scala v2.12  
([https://archive.apache.org/dist/kafka/2.4.1/kafka\\_2.12-2.4.1.tgz](https://archive.apache.org/dist/kafka/2.4.1/kafka_2.12-2.4.1.tgz)).
2. Un-tar it and `cd kafka_2.12-2.4.1`.
3. Start ZooKeeper.

```
1 bin/zookeeper-server-start.sh config/zookeeper.properties
```

4. Start Kafka broker.

```
1 bin/kafka-server-start.sh config/server.properties
```

5. Create topics.

```
1 bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic bochk
2 bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic hsbc
3 bin/kafka-topics.sh --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic hang-seng
```

## Deploy Third Party Bank Services (Require Python 3.7)

We take HSBC as an example.

1. Find `hsbc` directory in source codes and `cd` into it.

```
1 cd hsbc
```

2. (Optional) Run the following command to create a Python virtual environment. The second "venv" is the path to the new virtual environment, which is often set to "venv" directly, that is, the following command can be run directly without change.

```
1 python3 -m venv venv
```

### 3. Install Python requirements.

```
1 pip3 install -r requirements.txt
```

### 4. Run `main.py`.

```
1 python3 main.py
```

5. (Optional) If willing to deploy one more HSBC service instance, the simplest way is to change the port number in `main.py` and run `python3 main.py` again and the new instance will register to Nacos automatically. In practice, I will use Kubernetes in this kind of scenario to deploy multiple same instances, but to avoid introducing too many extra requirements, I refer to this approach in this instruction.

```
1 import os
2
3 # set environment variables
4 os.environ['FLASK_ENV'] = 'prod'
5 os.environ['FLASK_APP'] = 'hsbc'
6
7
8 from hsbc import app, bus
9 import api
10
11
12 if __name__ == '__main__':
13     bus.run()
14     app.run(port=5001)    # <----- HERE!!!!!!! -----
```

6. The procedures for HANG SENG and BOCHK are totally the same. They can be found in `hang_seng` and `bochk` directories respectively.

## Deploy BODB Backend (Require JDK 1.8+ and Maven)

1. Find `duck-bear-bank` directory in source codes and `cd` into it.
2. Run `mvn clean && mvn package` to build.
3. `cd target` and you may find `duck-bear-bank-0.0.1-SNAPSHOT.jar` in the `target` directory.

4. Run `java -jar duck-bear-bank-0.0.1-SNAPSHOT.jar` to start the server. It will run on port 8081 by default.

## Deploy BODB frontend (Require NodeJS 16+ and NPM)

1. Find `bodb` directory in source codes and `cd` into it.
2. Run `npm run install` to install requirements.
3. Run `npm run serve` to start the web server. By default it will use port 8080. Again, this approach is only for simplicity. In production mode, we may first build it into static files, and deploy it with Apache or Nginx.