

Kako je napravljeno?

Po preporuci za WebService uzeo sam Javu, odnosno Spring framework.

Iako nisam nešto iskusan u javi, uz proslijeđeni tutorial nije bilo strašno. Priznajem da ga nisam cijelog pogledao pa vjerojatno ima razlika i isto nije napravljeno po principima frameworka, ali meni je bilo važno pokriti funkcionalnosti zadane u zadatku.

1. WebService

Dakle, servis po sebi ima dvije važne komponente:

- 1.) API – omogućiti REST API i reagirati na njegove metode
- 2.) Background task – dohvaćanje podataka s Kraken API-a.

Unutar main funkcije webservisa pokreće se Spring aplikacija(po defaultu), te je ondje dodana još jedna linija koja će pokrenuti komponentu pod 2..

BackgroundScheduledJob - kreiran je kao Service Class, koju čine dvije metode te jedan privatni član.

- periodInMinutes – opisuje koliko često će se callback metoda izvršavati
- Start() – Metoda koja pokreće callback thread
- MainCallback() – Metoda koja se izvršava svakih „periodInMinutes“ minuta

Kraken

Unutar MainCallback() metode koristi se Kraken servis, koji je također definiran jednom Service Classom

- SourceAPIBaseURL – string koja je osnova kraken API URL-a
- PairNames – lista stringova koja sadrži moguće inpute za „PairName“ API key
- GetTickerInformation() – metoda koja za svaki mogući „PairName“ dohvaća podatke pomoću RestAPIWrapper.GET metode sa zadanog URL-a. Kreira novi objekt tipa Ticker, pridružuje vrijednosti za CreationTime, Data i PairName. U slučaju pogreške, ispisuje log.

RestAPIWrapper

Wrapper koji u ovom primjeru sadrži samo GET metodu(samo je ona potrebna).

- GET(string url) – metoda koja kreira GET request na zadani URL, primljeni response parsira u JSON Objekt koji vraća kao povratnu vrijednost. U slučaju pogreške, ispisuje log.

MemoryDataBaseService

Za početak(kasnije je prebačeno na postgres) DB je definirana u memoriji.

- DataBase – statički objekt tipa DataBase
- Insert(Ticker ticker) – dodaje Ticker u DB
- Select() – vraća listu svih Tickera u bazi
- Select(String pairName) - vraća listu svih Tickera u bazi s konkretnim pairNameom
- Select(String date, String pairName) - vraća listu svih Tickera u bazi s konkretnim pairNameom koji je nastao u zadanom vremenu – date.

Ticker – model podatka koji se provlači kroz cijelu aplikaciju

DataBase – primjer baze podataka u RAM-u, definirana kao lista objekata tipa Ticker.

TickerController – klasa unutar koje su definirani mogući API zahtjevi

```
8 @SpringBootApplication
9 public class Base58Application {
10     public static void main(String[] args) {
11         //run all SPRING services related to DB and API
12         SpringApplication.run(Base58Application.class, args);
13         //initializing scheduled task to fetch data from source API
14         BackgroundScheduledJob.Start();
15     }
16 }
```

Run: Base58Application

Console Endpoints

"C:\Program Files\Java\jdk-11.0.11\bin\java.exe" ...

```

  ____
 / \  / ___'  _ _ _ _ _ _ _ _ _ _ \
(  )\___| ' _ | ' _ | ' _ | ' _ |
 \ /  ___| | | | | | | | | | | | |
  '  |___| |_| | |_| | |_| | |_| |
=====|_|=====|_|_|_|_|_|_|_|_|
:: Spring Boot ::                (v2.4.5)

2021-05-16 15:14:01.725 INFO 13704 --- [main] com.example.demo.Base58Application : Starting Base58Application using Java 11.0.11 on LUKA-DESKTOP with PID 13704
2021-05-16 15:14:01.727 INFO 13704 --- [main] com.example.demo.Base58Application : No active profile set, falling back to default profiles: default
2021-05-16 15:14:02.589 INFO 13704 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2021-05-16 15:14:02.600 INFO 13704 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2021-05-16 15:14:02.600 INFO 13704 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.45]
2021-05-16 15:14:02.660 INFO 13704 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2021-05-16 15:14:02.660 INFO 13704 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 888 ms
2021-05-16 15:14:03.063 INFO 13704 --- [main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
2021-05-16 15:14:03.380 INFO 13704 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2021-05-16 15:14:03.392 INFO 13704 --- [main] com.example.demo.Base58Application : Started Base58Application in 2.002 seconds (JVM running for 2.825)
2021-05-16 15:14:03.463 INFO 13704 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2021-05-16 15:14:03.454 INFO 13704 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2021-05-16 15:14:03.454 INFO 13704 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 0 ms

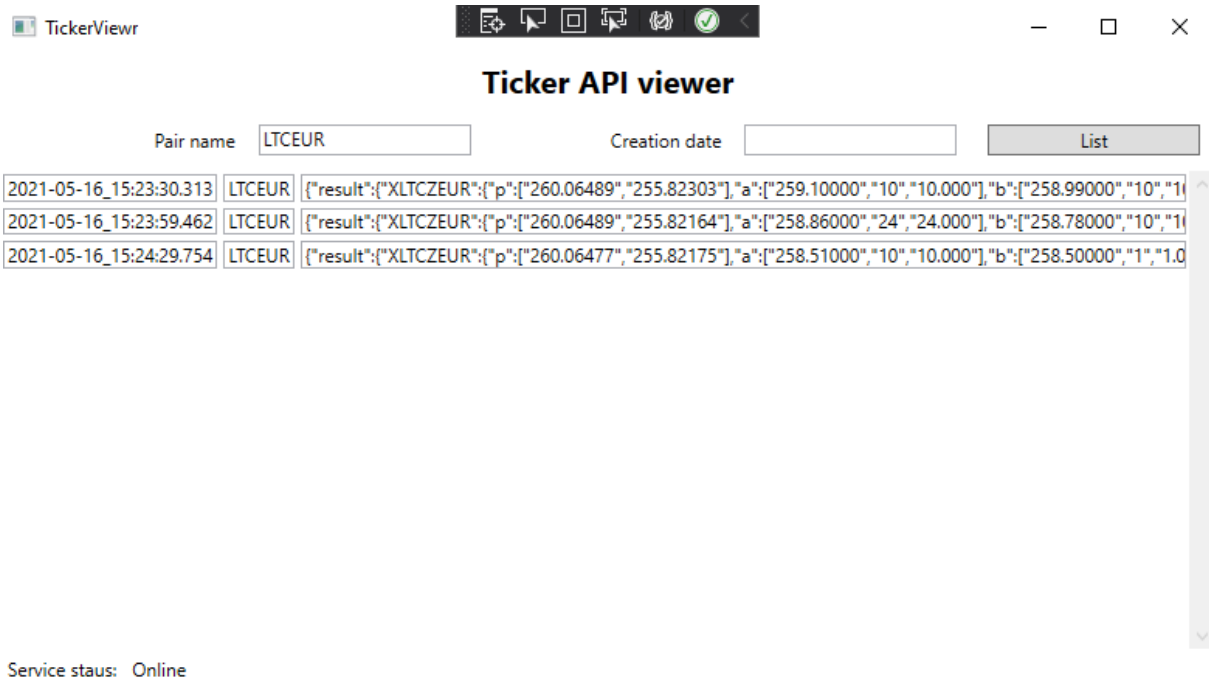
Ticker added to memory DB: 2021-05-16_15:14:04.142
Ticker added to memory DB: 2021-05-16_15:14:04.352
Ticker added to memory DB: 2021-05-16_15:14:04.394
Ticker added to memory DB: 2021-05-16_15:14:04.609
```

2. Consumer App

C# WPF aplikacija, kreiran po MVVM principu s popratnim Service i Common projektom.

Dizajniran je jednostavan prozor koji po pokretanju u donjem lijevom uglu pokazuje vidi li aplikacija webservice – ova se provjera događa u pozadini konstanto tako.

Dodano je i nekoliko kontrola kojima je moguće testirati API „(localhost:8080)/api/v1/ticker“



Kako pokrenuti?

Koliko sam shvatio spring framework što se tiče web servisa podigne sve sam čim se projekat pokrene unutar IDEA, pa je za sam servis dovoljno pokrenuti projekt.

*Baza podataka je podignuta pomoć dockera localhost:5432

Na strani Consumer App, ukoliko i nije instaliran MS Visual Studio, taj folder je pushan bez .gitignore filea, pa su i binary datoteke ondje, koje je dovoljno samo pokrenuti.

..\ConsumerApp\ConsumerApp\bin\Debug\ConsumerApp.exe