REFERENCE PROJECT

By:
Santosh Shitole

Project Title:
STOCKTREND

16th April 2017

TABLE OF CONTENTS

Project Description

The main aim of this project is to build a working application of moderate complexity to demonstrate the use of coding tools, libraries and technologies available as open source. For this purpose, the project aims to develop a tool to perform analysis on historical stock prices to determine potential buy / sell signals.

Project Goals

The project aims to demonstrate the use of the following techniques:

Code to Interfaces: Coding to interfaces is a technique by which developers can expose certain methods of an object to other objects in the system. The developers who receive implementations of these interfaces have the ability to code to the interface in place of coding to the object itself. In other words, the developers would write code that did not interact directly with an object as such, but rather with the implementation of that object's interface. Use of interfaces makes code loosely coupled and helps in defining responsibilities of a class. It also helps in writing more testable code.

Inversion of Control: Use of spring framework inversion of control. IoC provides decoupling components and layers in the system. Alleviates a component from being responsible for managing it's dependencies. Swap dependency implementations in different environments. Allows a component be tested through mocking of dependencies. Provides a mechanism for sharing resources throughout an application.

Using Yahoo API for gathering historical Data: Yahoo API provides historical data for most of the stocks. The project will use Yahoo API to gather historical data, for the given time period.

Using Esper CEP Engine: The project will demonstrate the use of Esper Complex Event Processing (CEP) engine to run stock analysis at run time. (Esper CEP. 2017).

Use of Gap Open Analysis: A price gap up or down in price can actually be a determination of the overall direction the stock will move in the coming months. A big price gap on very high volume, which means strong institutional buying of the stock, could mean more higher prices to come. The project will analyze to find gap open up of more than 2% over the previous close. Gap

open may occur due to some news on the stock fundamentals and may be considered as a Buy signal for long term investor. The project will build a framework to ease development and integration of new analysis components for future.

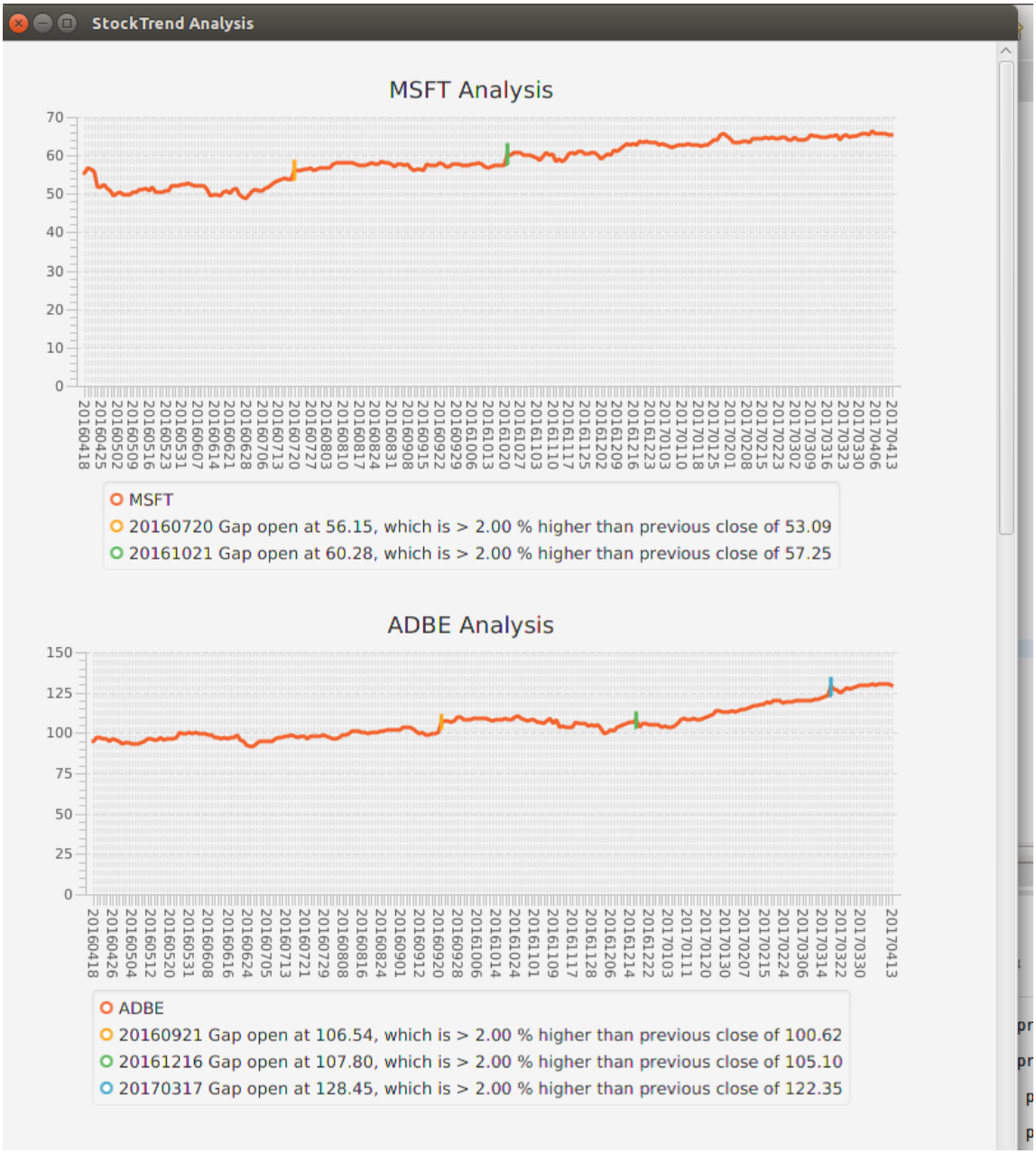
Use of JavaFx for creating stock charts: The project will use JavaFx API to create stock charts at run time. JavaFx is an evolving technology for developing polished, lightweight desktop applications.

Use of Maven: The project will use maven tool to develop project skeleton and to manage project dependencies (Apache Maven. 2017). It will also demonstrate JUnit tests using maven support for JUnit test cases.

UML Designing: The project will use ArgoUML (open source) to draw UML class diagram for important classes in the project. (ArgoUML. 2017)

Sample Run

The following screen shot displays result of a sample run. It displays stock chart for Microsoft(MSFT) and other stocks configured for the run. Gap open of more than 2% are highlighted on the chart. Gap open can be considered as Buy signal for a long term investor.

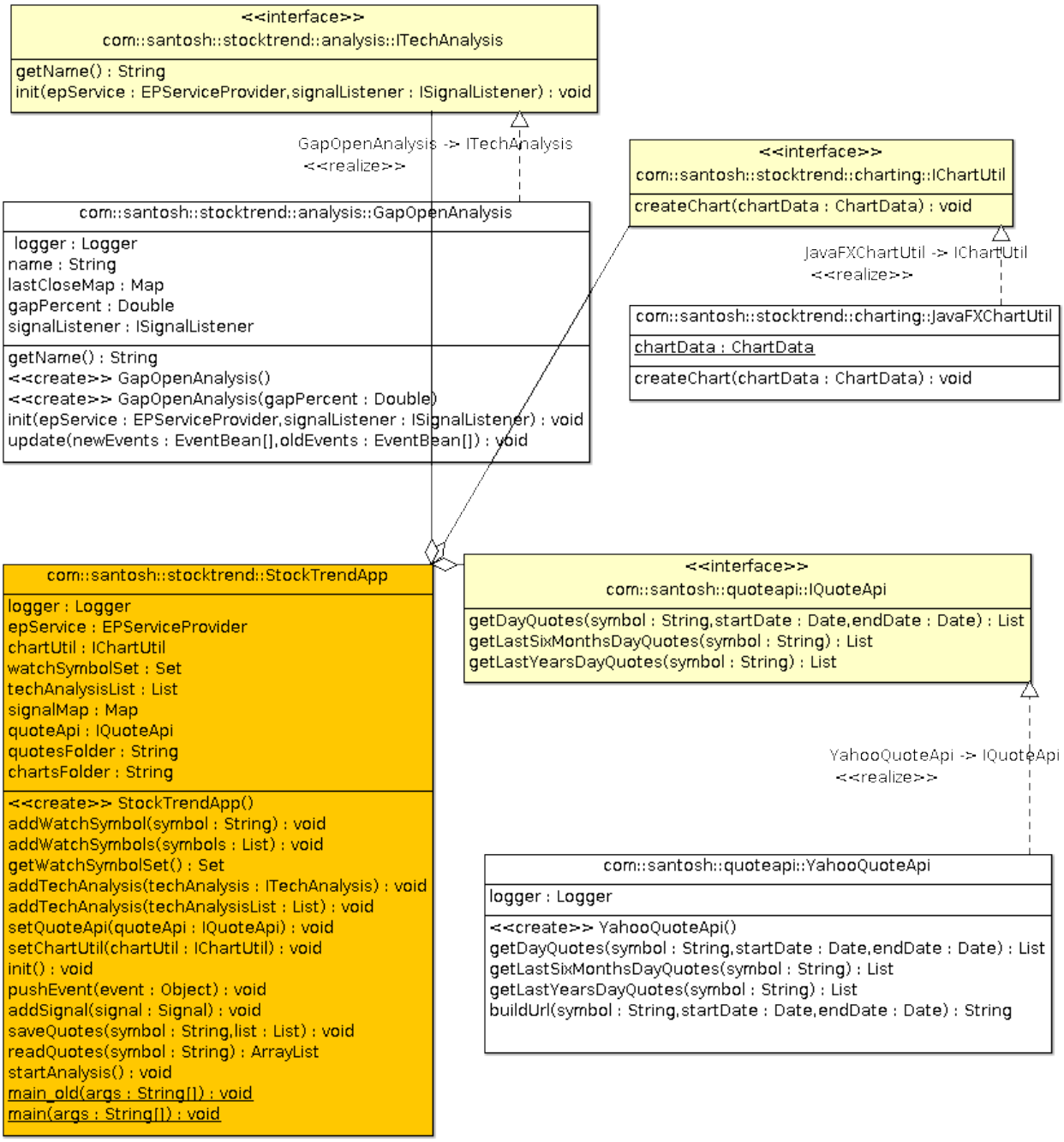


Project Work flow

The project application loads list of symbols provided in ‘watch symbol’ parameter list. New symbols can be added in without the need of code compilation.
The application connects to Yahoo API and retrieves historical price information for the previous year.
It instantiates an instance of Esper CEP engine and sends price events in to the engine.
Technical analysis components receive price events. GapOpenTechnicalAnalysis components compares open price with the previous close. If the gap open is more than 2% up, then it creates a signal.
After all price events are processed, JavaFxChartingUtil component creates price charts and tech analysis signals.

Class Diagram

Class diagram shows relationship of important classes and interfaces within the project space.
(Note: Please expand the diagram for better view)



Inversion Of Control

The project demonstrates the use of spring framework IoC. Old way uses component integration

in the main method, as:

```
        public static void main_old( String[] args ){

                StockTrendApp app = new StockTrendApp();
        // add watch symbols
        app.addWatchSymbol("SWKS");
        app.addWatchSymbol("AAPL");
        app.addWatchSymbol("MSFT");
        app.addWatchSymbol("ADBE");
        app.addWatchSymbol("AMZN");

                // conventional way of wiring dependencies
          app.chartUtil = new JavaFXChartUtil();
        app.addTechAnalysis(new GapOpenAnalysis());
        app.quoteApi = new YahooQuoteApi();
        app.startAnalysis();
    }
```

Spring IoC uses wiring defined in Spring-Module.xml, to achieve the same.

```
<beans xmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-4.2.xsd">

        <bean id="yahooQuoteApi" class="com.santosh.quoteapi.YahooQuoteApi">
        </bean>

        <bean id="gapOpenTA" class="com.santosh.stocktrend.analysis.GapOpenAnalysis">
        </bean>

        <bean id="javaFxChartUtil" class="com.santosh.stocktrend.charting.JavaFXChartUtil">
        </bean>

        <bean id="app" class="com.santosh.stocktrend.StockTrendApp">
           <property name="quoteApi" ref="yahooQuoteApi"/>
           <property name="chartUtil" ref="javaFxChartUtil"/>
        </bean>

        <bean class="org.springframework.beans.factory.config.MethodInvokingBean">
           <property name="targetObject" ref="app"/>
           <property name="targetMethod"><value>addTechAnalysis</value></property>
           <property name="arguments">
         <list>
                 <ref bean="gapOpenTA"/>
         </list>
           </property>
        </bean>

        <bean class="org.springframework.beans.factory.config.MethodInvokingBean">
           <property name="targetObject" ref="app"/>
           <property name="targetMethod"><value>addWatchSymbols</value></property>
           <property name="arguments">
         <list>
             <value>AMZN</value>
             <value>ADBE</value>
             <value>MSFT</value>
             <value>ORCL</value>
         </list>
```

```
        </property>
       </bean>

       <bean id="startAnalysis"
class="org.springframework.beans.factory.config.MethodInvokingBean">
          <property name="targetObject" ref="app"/>
          <property name="targetMethod"><value>startAnalysis</value></property>

       </bean>


</beans>
```

Evaluation

StockTrend uses EsperCEP engine to process price events at runtime. Following are the sample logs generated when the project was configured for watch symbol list of AMZN, ADBE, MSFT and ORCL. Total 4 symbols resulted in generation of 1004 (4 * 251) events. These events were processed within 3 seconds.

Sample Logs:

```
2017-04-16 19:50:21,838 org.springframework.context.support.ClassPathXmlApplicationContext INFO
- Refreshing org.spring
framework.context.support.ClassPathXmlApplicationContext@20fa23c1: startup date [Sun Apr 16
19:50:21 EDT 2017]; root of
context hierarchy
2017-04-16 19:50:21,884 org.springframework.beans.factory.xml.XmlBeanDefinitionReader INFO  -
Loading XML bean definitio
ns from class path resource [Spring-Module.xml]
2017-04-16 19:50:22,159 com.espertech.esper.core.service.EPServiceProviderImpl INFO  -
Initializing engine URI 'default'
 version 5.3.0
2017-04-16 19:50:24,077 StockTrendApp INFO  - 20160720 Gap open at 56.15, which is > 2.00 %
higher than previous close o
f 53.09
2017-04-16 19:50:24,077 StockTrendApp INFO  - 20161021 Gap open at 60.28, which is > 2.00 %
higher than previous close o
f 57.25
2017-04-16 19:50:24,096 StockTrendApp INFO  - 20160921 Gap open at 106.54, which is > 2.00 %
higher than previous close
of 100.62
2017-04-16 19:50:24,096 StockTrendApp INFO  - 20161216 Gap open at 107.80, which is > 2.00 %
higher than previous close
of 105.10
2017-04-16 19:50:24,096 StockTrendApp INFO  - 20170317 Gap open at 128.45, which is > 2.00 %
higher than previous close
of 122.35
2017-04-16 19:50:24,102 StockTrendApp INFO  - 20160617 Gap open at 39.49, which is > 2.00 %
higher than previous close o
f 38.64
2017-04-16 19:50:24,102 StockTrendApp INFO  - 20170316 Gap open at 46.39, which is > 2.00 %
higher than previous close o
f 43.05
2017-04-16 19:50:24,106 StockTrendApp INFO  - 20160429 Gap open at 666.00, which is > 2.00 %
higher than previous close
of 602.00
2017-04-16 19:50:24,106 StockTrendApp INFO  - 20160510 Gap open at 694.00, which is > 2.00 %
higher than previous close
of 679.75
2017-04-16 19:50:24,106 StockTrendApp INFO  - 20161107 Gap open at 771.64, which is > 2.00 %
higher than previous close
of 755.05
```

REFERENCES

Apache Maven. 2017. Apache Maven Project. Retrieved on 16th April 2017 from http://maven.apache.org/

ArgoUML. 2017. Tigris Open Source Software Engineering Tools. Retrieved on 16th April 2017 from http://argouml.tigris.org/

Esper CEP. 2017. Esper Complex Event Processing. Retrieved on 16th April 2017 from http://www.espertech.com/esper/