Seminars in Distributed Systems
a.a. 2010/2011
Monitoring Financial Market Indexes

Professor: Roberto Baldoni

Instructor: Luca Montanari

Student: **Antonio Paolacci**

The aim of the work (1)

Historical data have shown that several patterns in the behavior of the stock market are repeatable and thus they can be recognized and used to predict the future behavior of the markets. The aim of this work is to collect stock market data and trigger alerts if a given pattern is discovered at runtime. In order to present the work, the application has to work also offline using log-files.

The aim of the work (1.1)

To get friendly with:

- **CEP** (Complex Event Processing) technologies and
- the **ESPER** open-source java framework

Examples of use of these could be:

- disseminate the right information to the right receiver,
- to do active diagnostics observing sub-sequent events,
- to prevent IT-failures and
- to detect malicious behaviors like an intrusion detection system.

Esper

It offers:

- a specific language for events definitions(**EPL**) and
- a specific engine to detect events and pass them to the listeners class

- The main concept: **time** and batch/length windows. Data flow through the long running queries, no persistence.

Some typical examples of Esper applications are:

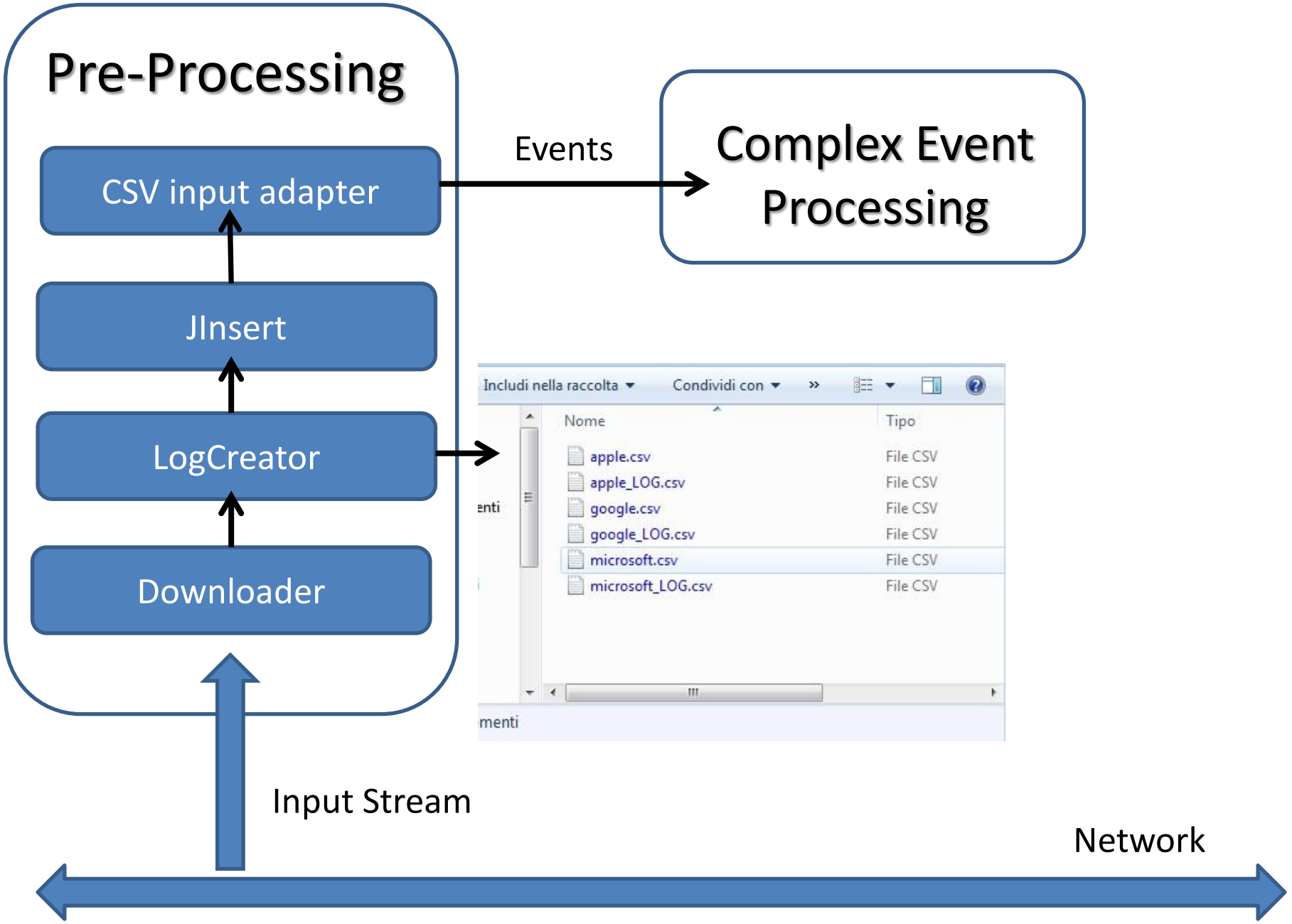
- **finance** (algorithmic trading, fraud detection),
- **network and application monitoring** (intrusion detection system, SLA monitoring),
- **sensor network applications** (RFID, scheduling and control of fabrication lines, air traffic control),
- **tv programs application** (televoting control and call-center fraud).

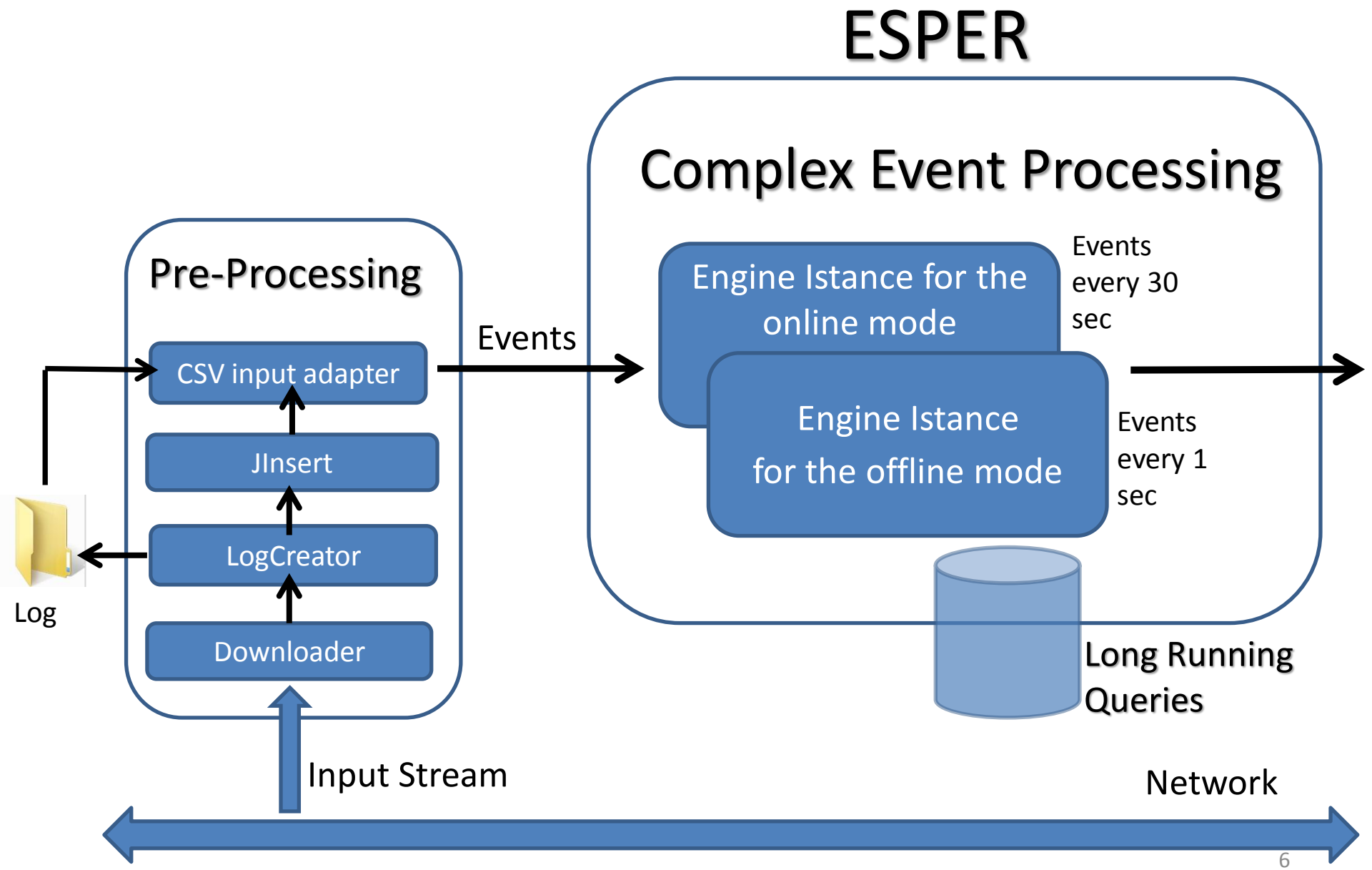
The implemented tool

This presentation describes a tool designed to support stock market investors during their analysis of stock quotes using ESPER. The system operates on real-time stock quotes, downloaded from Yahoo!Finance web service in CSV files.

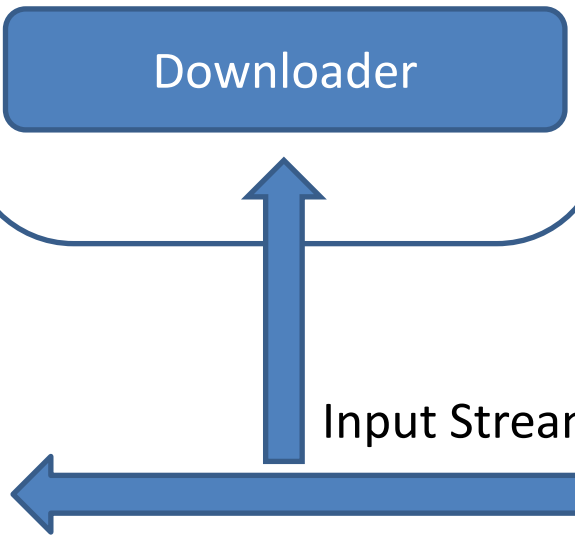
Main features :

- the use of **CSV input adapter** that reads comma separated values and sends event to the Esper engine;
- the use of **CSV input adapter coordinator** to simulate multiple streams of events;
- some Esper procedure for the disclosure of "**all indexes are down**" , "**more than 2 indexes in your portfolio are down of almost 1% point**" and an economic pattern called "**Triple Bottom Pattern**";
- the development of a **GUI** which can effectively support investors in their activities.

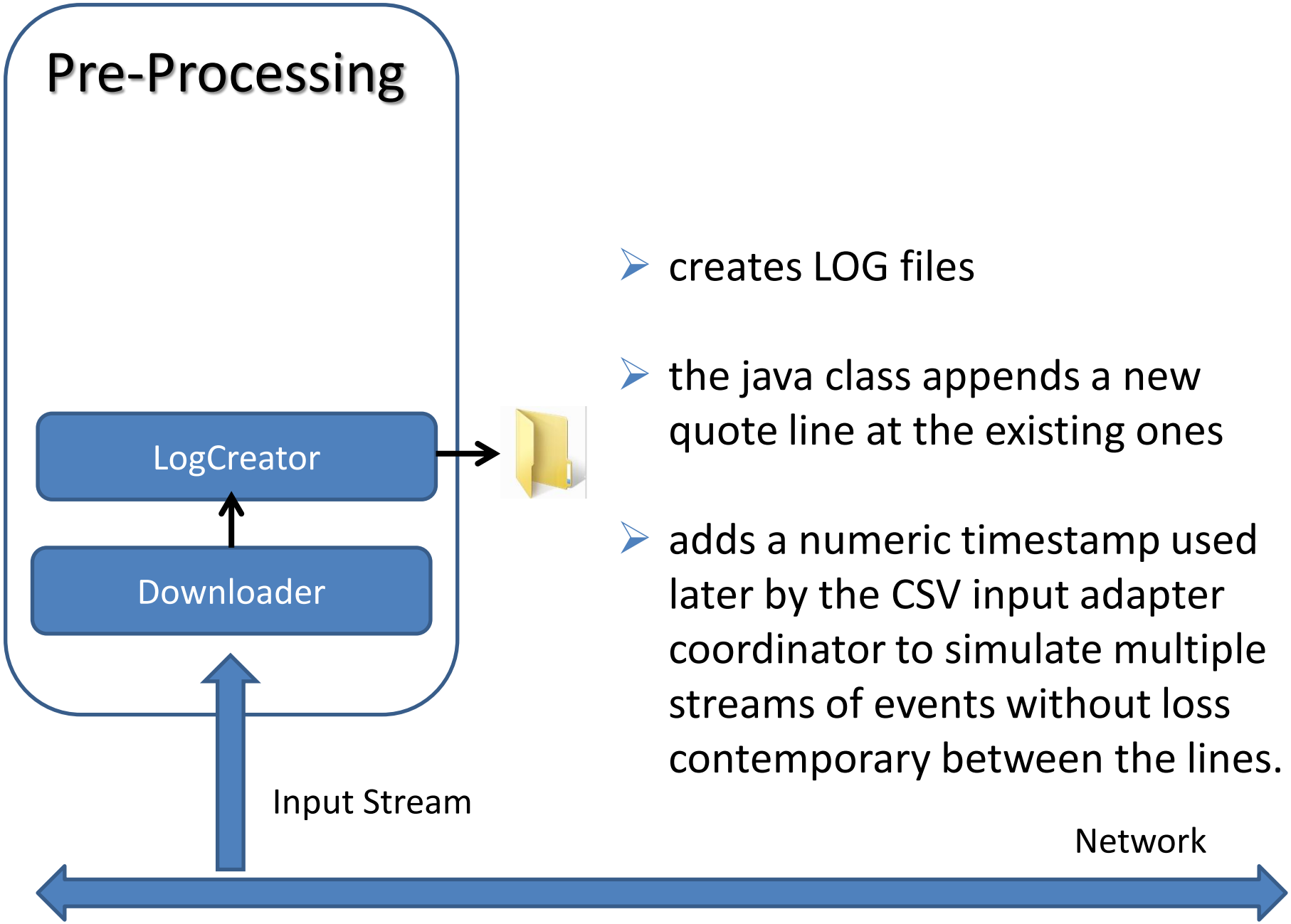

Pre-Processing
CSV input adapter
JInsert
LogCreator
Downloader
Events
Complex Event Processing
Includi nella raccolta
Condividi con
Nome
Tipo
apple.csv
File CSV
apple_LOG.csv
File CSV
google.csv
File CSV
google_LOG.csv
File CSV
microsoft.csv
File CSV
microsoft_LOG.csv
File CSV
Input Stream
Network


ESPER
Complex Event Processing
Pre-Processing
CSV input adapter
JInsert
LogCreator
Downloader
Log
Input Stream
Events
Engine Istance for the online mode
Events every 30 sec
Engine Istance for the offline mode
Events every 1 sec
Long Running Queries
Network

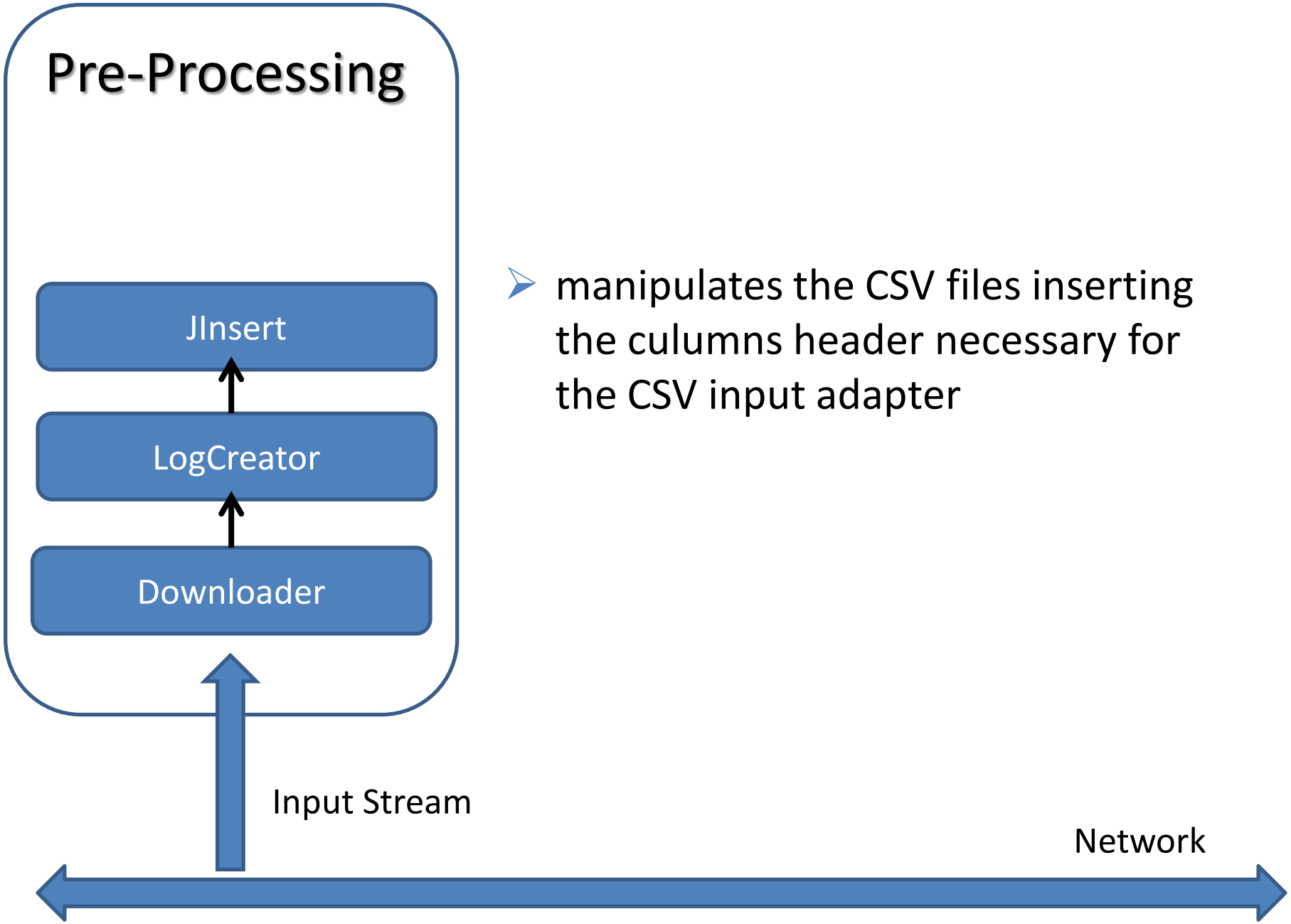
Pre-Processing



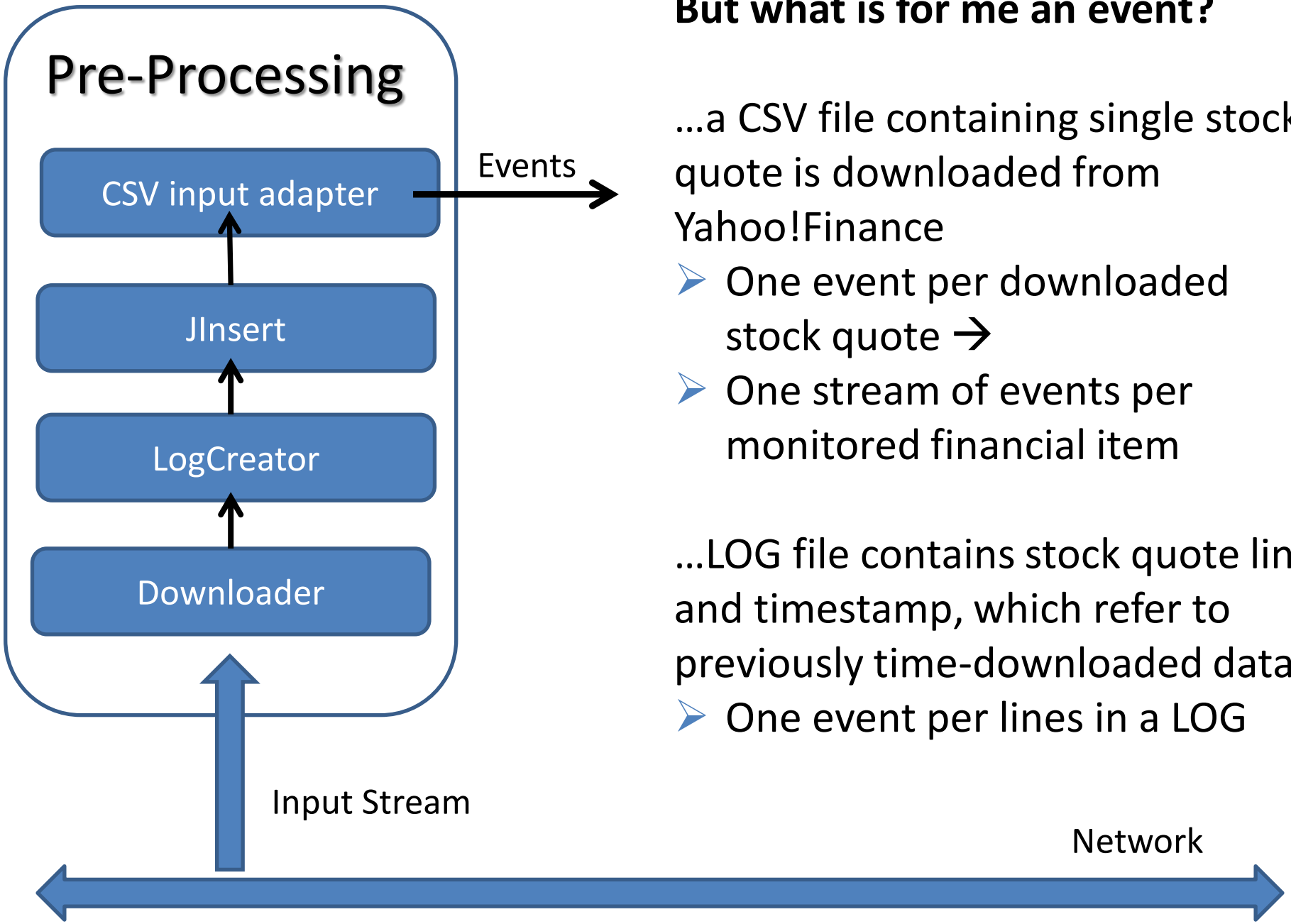
- downloads real time stock quotes about Apple,Google and Microsoft separately quering Yahoo!Finance web service at the following address http://finance.yahoo.com/d/quotes.csv?s=AAPL&f=sl1c6d1t1
- the server responds with a CSV file which contains a single quote line
- The CSV file overwrites the previous downloaded file.



- creates LOG files
- the java class appends a new quote line at the existing ones
- adds a numeric timestamp used later by the CSV input adapter coordinator to simulate multiple streams of events without loss contemporary between the lines.



- manipulates the CSV files inserting the culumns header necessary for the CSV input adapter



But what is for me an event?

…a CSV file containing single stock quote is downloaded from Yahoo!Finance

- One event per downloaded stock quote →
- One stream of events per monitored financial item

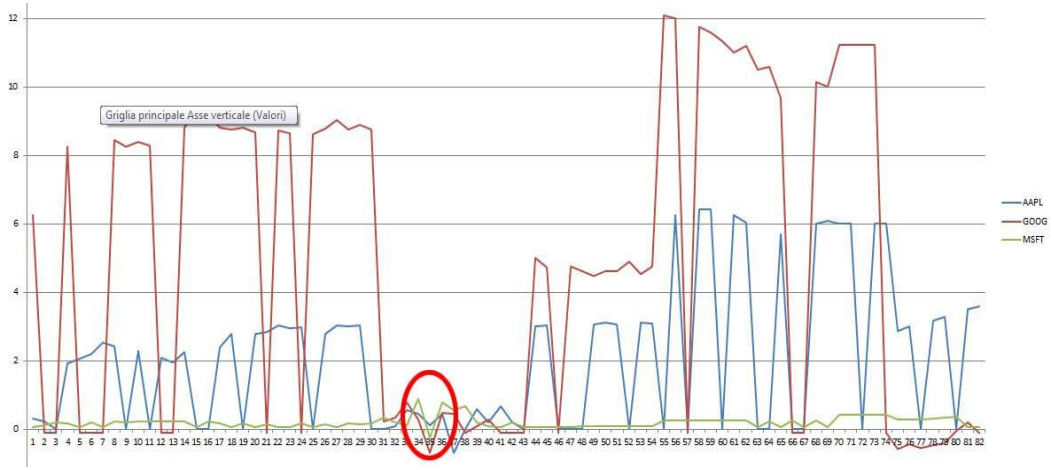
…LOG file contains stock quote lines and timestamp, which refer to previously time-downloaded data

- One event per lines in a LOG

The long Running Queries

All indexes in your portfolio are down at current time.

- Time windows
- Events correlation




```
/*the EPL statement*/
String expression4 = "select apple.itemName, apple.lastDate, apple.lastTime,
google.itemName, microsoft.itemName
from AAPLevent.win:time(30 sec) as apple, GOOGevent.win:time(30 sec) as google,
MSFTevent.win:time(30 sec) as microsoft
where apple.change<=0 and google.change<=0 and microsoft.change<=0";

EPStatement statement4 = epService.getEPAdministrator().createEPL(expression4);
statement4.addListener(listener2);

/*the listener class associated to the event*/
public class MyListener2 implements UpdateListener {

    public void update(EventBean[] newEvents, EventBean[] oldEvents) {
```

Alert if more than 2 indexes are down of almost one point %.

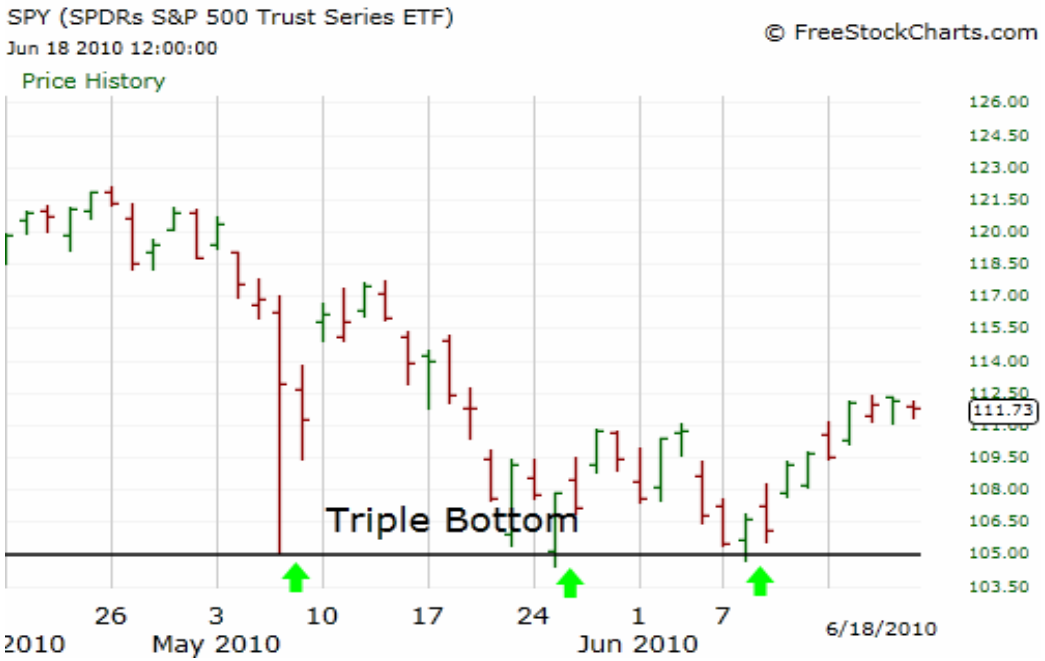
- named windows

```
EPStatement statement5 = epService.getEPAdministrator().createEPL("create window
 W1.win:time(30 sec) (item String)");
EPStatement statement6 = epService.getEPAdministrator().createEPL("insert into W1
select itemName as
item from AAPLevent.win:time(30 sec) as apple where apple.change<=-1");
EPStatement statement7 = epService.getEPAdministrator().createEPL("insert into W1
select itemName as
item from GOOGevent.win:time(30 sec) as google where google.change<=-1");
EPStatement statement8 = epService.getEPAdministrator().createEPL("insert into W1
select itemName as
item from MSFTevent.win:time(30 sec) as microsoft where microsoft.change<=-1");
EPStatement statement9 = epService.getEPAdministrator().createEPL("select count(*)
as Nindici from W1 having count(*) = 3");

statement9.addListener(new MyListener3());
```

Triple Bottom Pattern
is composed of 3 lows, all at about the same price level.

Important things for the validity:

- at the beginning there is a downward trend in a stock's price history
- wait a breakout price point as a confirmation point



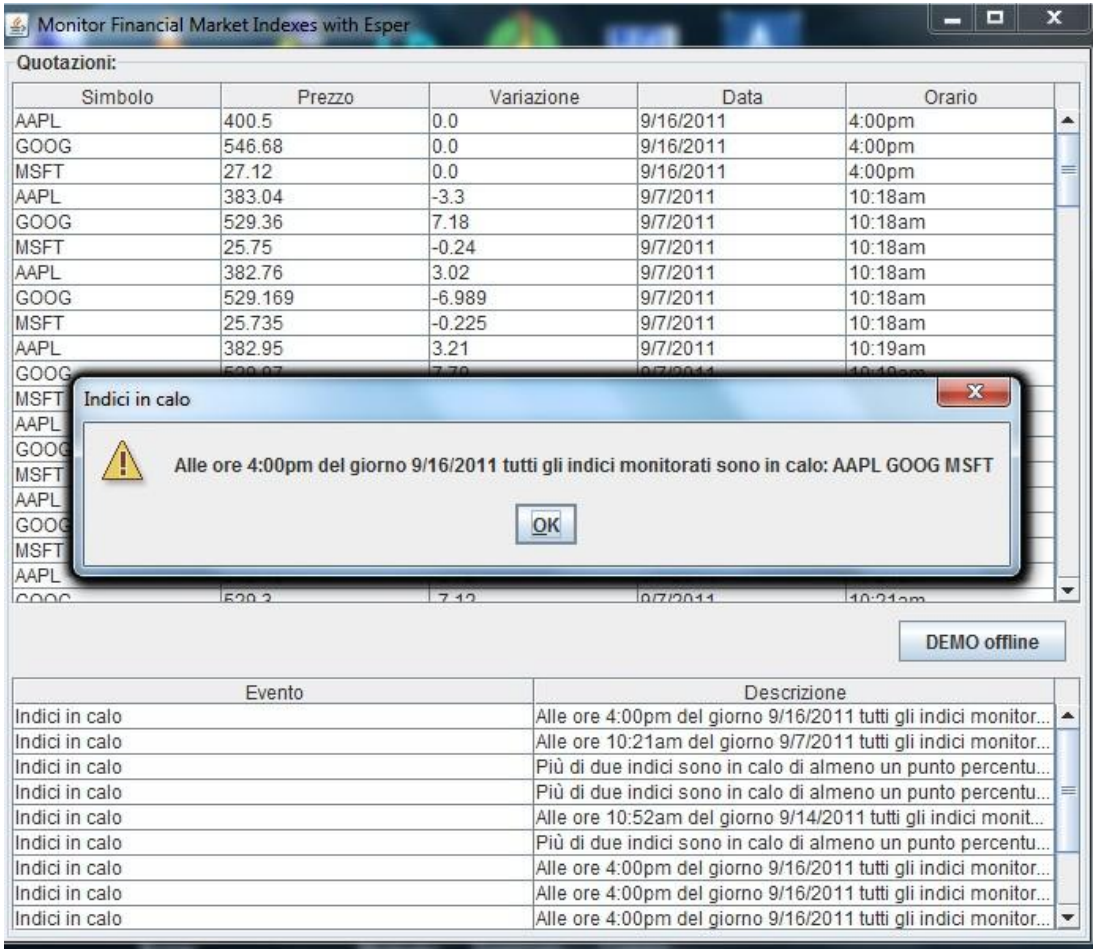
To detect a triple bottom pattern I need to use the EPL **pattern clause**.

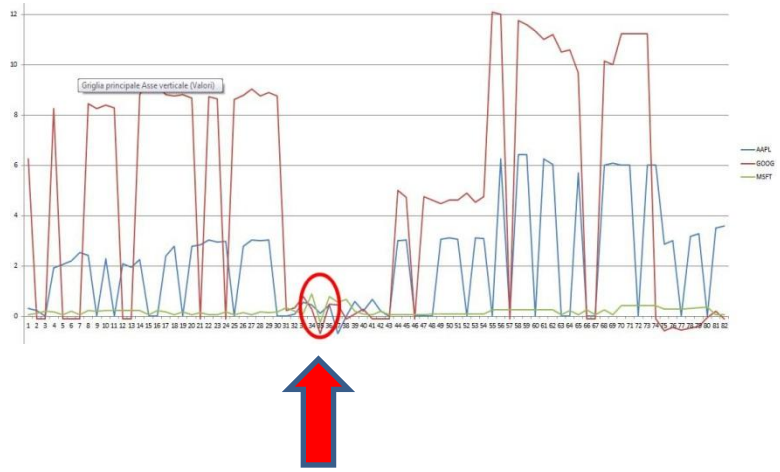
The algorithm

for *i = 0* **to** *10min* **do**
calculate avg(price) of GOOGevent
insert avg(price) into LastAvg10Min
output result every 30sec
end for
for *i = 0* **to** *5min* **do**
calculate avg(price) and min(price) of GOOGevent
insert avg(price), minPrice into LastAvg5Min
output result every 30sec
end for
if *LastAvg10Min:avgPrice * 0,85 > LastAvg5Min:avgPrice* **then**
insert minPrice into BottomPriceEvent
output result every 1min
end if
if *a BottomPriceEvent -> interval 3min -> a BottomP riceEvent(between 0,95 and 1,05 of previous) whithin 5min*
-> interval 3min -> a BottomPriceEvent (between 0,95 and 1,05 of previous) **then**
triple bottom pattern detected
end if

```
/*This query determines the average price for the last 10 minutes:*/
EPStatement statement10 = epService.getEPAdministrator().createEPL("insert into
 AvgPriceLast10Min
select avg(price) as avgPrice from GOOGevent.win:time(10 min)
output every 30 sec");

/*This query to determines the average price for the last 5 minutes:*/
EPStatement statement11 = epService.getEPAdministrator().createEPL("insert into
 AvgPriceLast5Min
select avg(price) as avgPrice, min(price) as minPrice
from GOOGevent.win:time(5 min)
output every 30 sec");

/*Compare the last average prices for each:*/
EPStatement statement12 = epService.getEPAdministrator().createEPL("insert into
 BottomPriceEvent
select minPrice as bottomPrice
from AvgPriceLast10Min.std:lastevent() as LastAvg10Min,
AvgPriceLast5Min.std:lastevent() as LastAvg5Min
where LastAvg10Min.avgPrice * 0.85 > LastAvg5Min.avgPrice
output first every 1 min");

/*the last query detect a Triple Bottom Pattern event*/
EPStatement statement13 = epService.getEPAdministrator().createEPL("insert into
 TripeBottomPattern
select * from pattern [every a=BottomPriceEvent -> timer:interval(3 min)
 -> BottomPriceEvent(bottomPrice between 0.95*a.bottomPrice and 1.05*a.bottomPrice)
where timer:within(5 min)
-> timer:interval(3 min) -> BottomPriceEvent(bottomPrice between 0.95*a.bottomPrice
and 1.05*a.bottomPrice) where timer:within(5 min)]");

statement13.addListener(new MyListener4());
```

Demo and…


Monitor Financial Market Indexes with Esper
Quotazioni:
Simbolo | Prezzo | Variazione | Data | Orario
AAPL | 400.5 | 0.0 | 9/16/2011 | 4:00pm
GOOG | 546.68 | 0.0 | 9/16/2011 | 4:00pm
MSFT | 27.12 | 0.0 | 9/16/2011 | 4:00pm
AAPL | 383.04 | -3.3 | 9/7/2011 | 10:18am
GOOG | 529.36 | 7.18 | 9/7/2011 | 10:18am
MSFT | 25.75 | -0.24 | 9/7/2011 | 10:18am
AAPL | 382.76 | 3.02 | 9/7/2011 | 10:18am
GOOG | 529.169 | -6.989 | 9/7/2011 | 10:18am
MSFT | 25.735 | -0.225 | 9/7/2011 | 10:18am
AAPL | 382.95 | 3.21 | 9/7/2011 | 10:19am
Indici in calo
Alle ore 4:00pm del giorno 9/16/2011 tutti gli indici monitorati sono in calo: AAPL GOOG MSFT
OK
DEMO offline
Evento | Descrizione
Indici in calo | Alle ore 4:00pm del giorno 9/16/2011 tutti gli indici monitor...
Indici in calo | Alle ore 10:21am del giorno 9/7/2011 tutti gli indici monitor...
Indici in calo | Più di due indici sono in calo di almeno un punto percentu...
Indici in calo | Più di due indici sono in calo di almeno un punto percentu...
Indici in calo | Alle ore 10:52am del giorno 9/14/2011 tutti gli indici monit...
Indici in calo | Più di due indici sono in calo di almeno un punto percentu...
Indici in calo | Alle ore 4:00pm del giorno 9/16/2011 tutti gli indici monitor...
Indici in calo | Alle ore 4:00pm del giorno 9/16/2011 tutti gli indici monitor...
Indici in calo | Alle ore 4:00pm del giorno 9/16/2011 tutti gli indici monitor...



...concluding remarks

- Yahoo!Finance web service: **free**, **easy** to use, **compatible** and it doesn't required an account or registration.

- to **use the CSV input adapter/coordinator** pay attention for some tricks:
  - java class with setter-methods and no constructor
  - to use POJO as events simply register the name for the Java class in configuration phase and provide the same name to the adapter
  - column header in the CSV is required unless a property order is defined explicitly.

- TripleBottomPattern queries are **long-running continuous queries** since they need to run **over days and month** → very difficult to simulate a TBP event in my demo

- **is an educational software**, simply gives an alert instead of (for e.g.) automatically executing trades