■ Flume으로 데이터를 수집해 HBase로 데이터 적재

- 요구사항

○ CSV 파일은 쉼표로 값이 구분되어 있음

○ 쉼표로 구분된 각각의 값을 하나의 컬럼으로 적재해야함

```
1010101                 column=location:city, timestamp=1529469708315, value=서울
1010101                 column=location:country, timestamp=1529469706772, value=한국
1010101                 column=message:message, timestamp=1529469698762, value=안녕하세요
1010101                 column=message:name, timestamp=1529469691283, value=김필구
```

▸ 옳은 예

```
1010101                 column=location:city, timestamp=1529469708315, value=서울,한국,안녕하세요,김필구
```

▸ 옳지 못한 예

○ CSV 파일의 데이터들을 컴마 단위로 파싱을 한 뒤 HBase에 데이터를 전송 해야함 -> serializer(직렬화)를 사용

○ Serializer(직렬화): 시스템 내부에서 사용되는 객체 또는 데이터를 외부의 시스템에서도 이용할 수 있도록 바이트(Byte) 형태로 데이터를 변환하는 기술

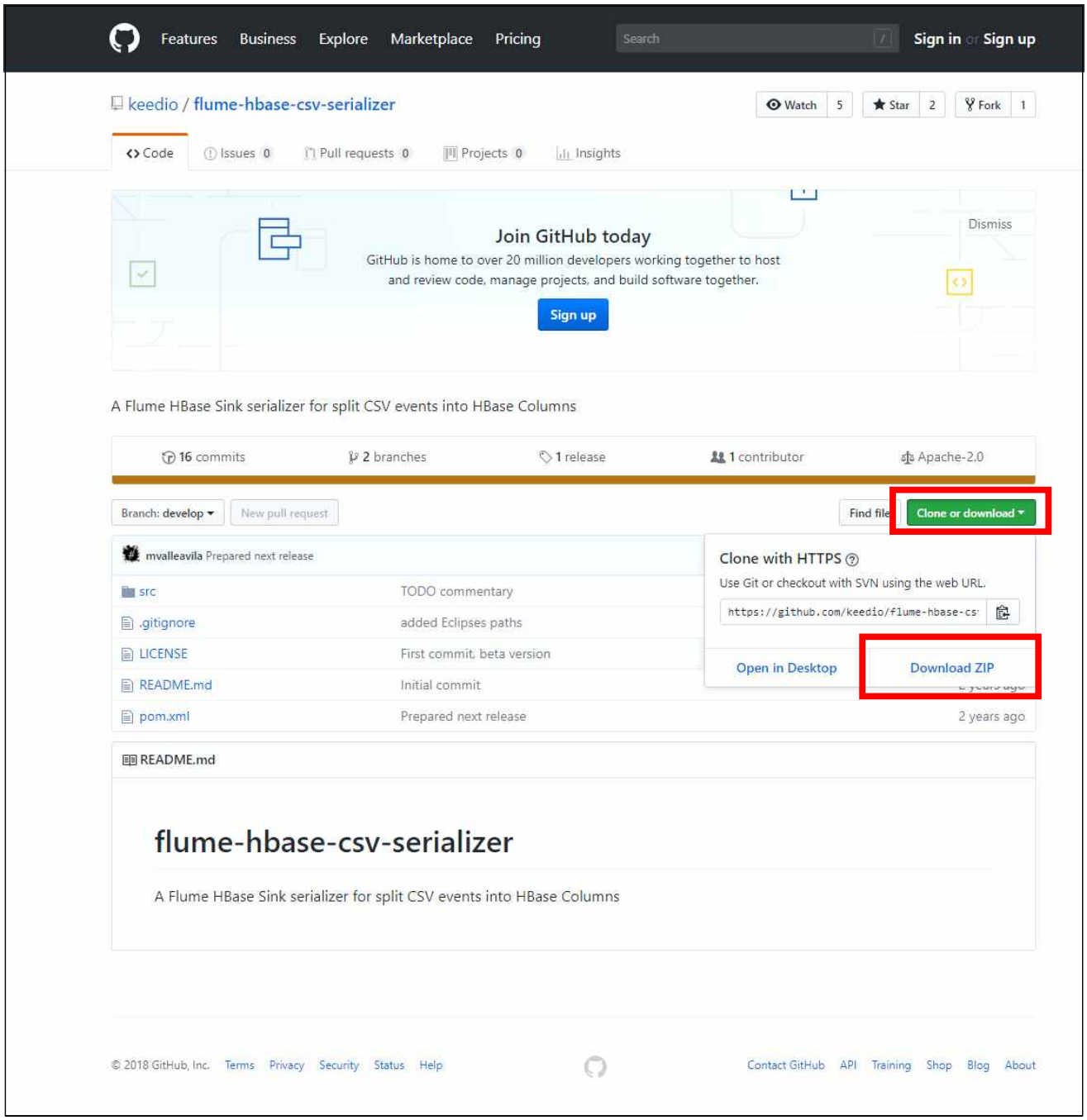
○ Flume에서 제공하는 직렬화 라이브러리 사용 하거나 직접 만들어서 사용

○ 우리는 keedio라는 회사에서 만든 Hbase용 csv 직렬화 라이브러리 사용

- 아래의 주소로 들어감

```
https://github.com/keedio/flume-hbase-csv-serializer
```

- Clone or download 버튼 클릭 후 Download ZIP 클릭



- 줄 바꿈 개행이 포함된 CSV 파일을 hbase로 적재할 때 줄 바꿈 개행값도 저장됨
- 만약 줄 바꿈 개행이 포함되어 있다면 삭제한 뒤 적재
- 압축을 풀고 CsvSerializer.java 파일을 열기

- getActions 메서드를 수정(136번째 줄)

```
  try {
    rowKey = RowKeyGenerator.generateRowKey(keyType);

    Put put = new Put(rowKey);

    for (int i=0; i<columnNames.size();i++){
      put.addColumn(cf, columnNames.get(i).getBytes(Charsets.UTF_8),
              eventSplitted[i].getBytes(Charsets.UTF_8));
    }
    actions.add(put);
```

```
try {
rowKey = RowKeyGenerator.generateRowKey(keyType);

Put put = new Put(rowKey);

for (int i=0; i<columnNames.size();i++){

byte[] events = eventSplitted[i].getBytes(Charsets.UTF_8);

if(i==columnNames.size()-1 && events[events.length-1]==13) {

byte[] tmp = new byte[events.length-1];

for (int j = 0 ; j < tmp.length ;j++) {
tmp[j] = events[j];
}
events = tmp;
}

put.addColumn(cf, columnNames.get(i).getBytes(Charsets.UTF_8),events);
}
actions.add(put);
```

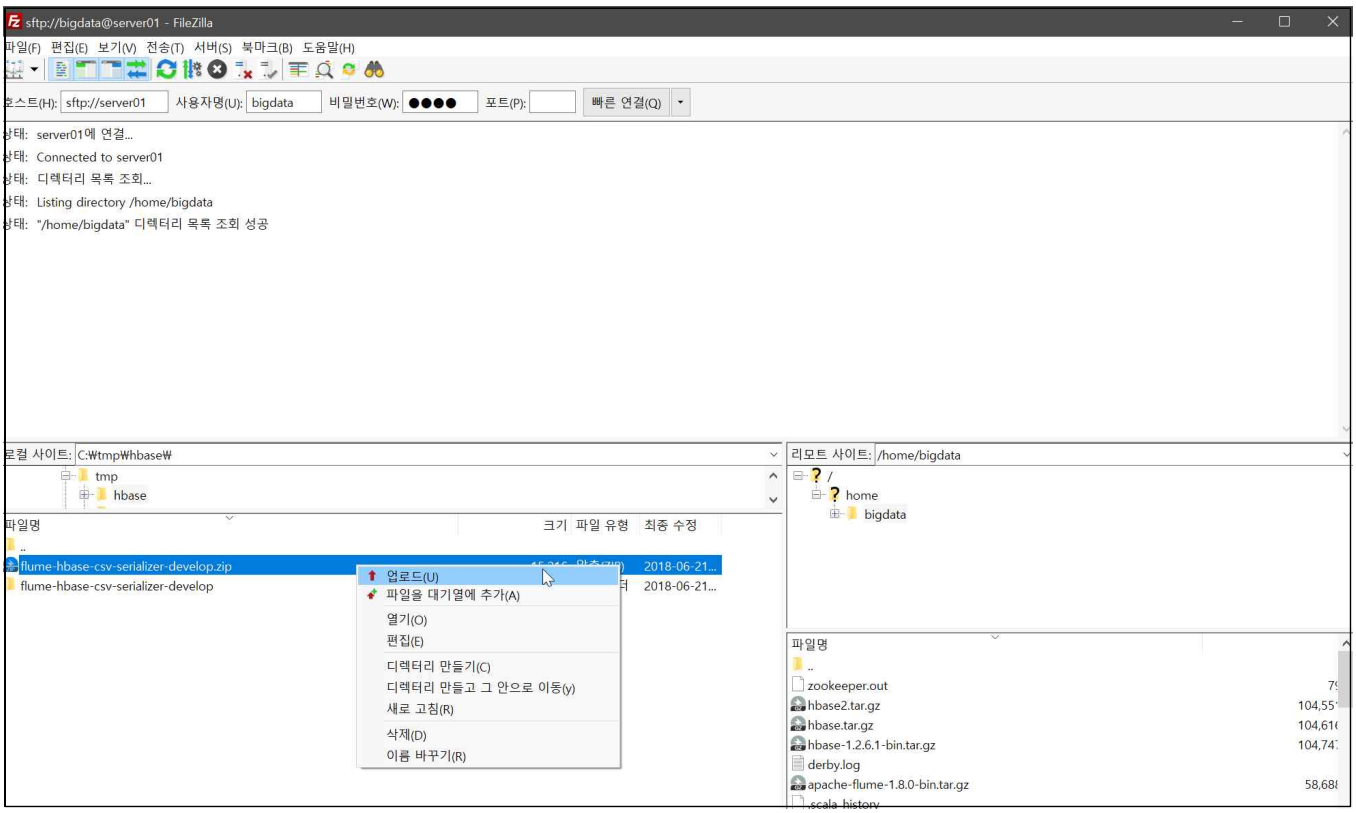
- getIncrements 메서드 안의 내용을 비움

```
@Override
public List<Increment> getIncrements(){
  List<Increment> increments = new LinkedList<Increment>();
  if(incCol != null) {
    Increment inc = new Increment(incrementRowKey);
    inc.addColumn(cf, incCol, 1);
    increments.add(inc);
  }
  return increments;
}
```

```
public List<Increment> getIncrements(){
List<Increment> increments = new LinkedList<Increment>();

return increments;
}
```

- 파일을 다시 압축한 뒤 FileZilla를 통해 Server01의 bigdata 홈디렉터리로 전송



- Server01에 root로 접속해 zip 압축 파일을 풀기 위한 unzip 설치

```
# yum -y unzip
```

- Server01에 bigdata계정으로 접속해 파일의 압축을 품

```
$ unzip flume-hbase-csv-serializer-develop.zip
```

- 압축푼 파일로 이동해 mvn package로 빌드 후 패키지 파일 생성

```
$ cd flume-hbase-csv-serializer-develop
```

```
$ mvn package
```

```
[WARNING] Javadoc Warnings
[WARNING] /home/bigdata/flume-hbase-csv-serializer-develop/src/main/java/org/kee
dio/flume/sink/hbase/serializer/CsvSerializer.java:59: warning: empty <p> tag
[WARNING] * an exception will be throw.<p>
[WARNING] ^
[INFO] Building jar: /home/bigdata/flume-hbase-csv-serializer-develop/target/csv
Serializer-0.0.2-SNAPSHOT-javadoc.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 01:43 min
[INFO] Finished at: 2018-06-21T11:12:21+09:00
[INFO]
```

- 성공했으면 target 디렉터리로 이동해 csvSerializer-0.0.2-SNAPSHOT.jar 확인

```
$ cd target/
$ ls
```

```
[bigdata@server01 target]$ ls
apidocs
classes
csvSerializer-0.0.2-SNAPSHOT-javadoc.jar
csvSerializer-0.0.2-SNAPSHOT-sources.jar
csvSerializer-0.0.2-SNAPSHOT.jar
generated-sources
generated-test-sources
javadoc-bundle-options
maven-archiver
original-csvSerializer-0.0.2-SNAPSHOT.jar
surefire-reports
test-classes
```

- Flume 디렉터리의 lib 디렉터리로 jar 파일 복사

```
$ cp csvSerializer-0.0.2-SNAPSHOT.jar /home/bigdata/apache-flume-1.8.0-bin/lib/
```

- Zookeepr 시작

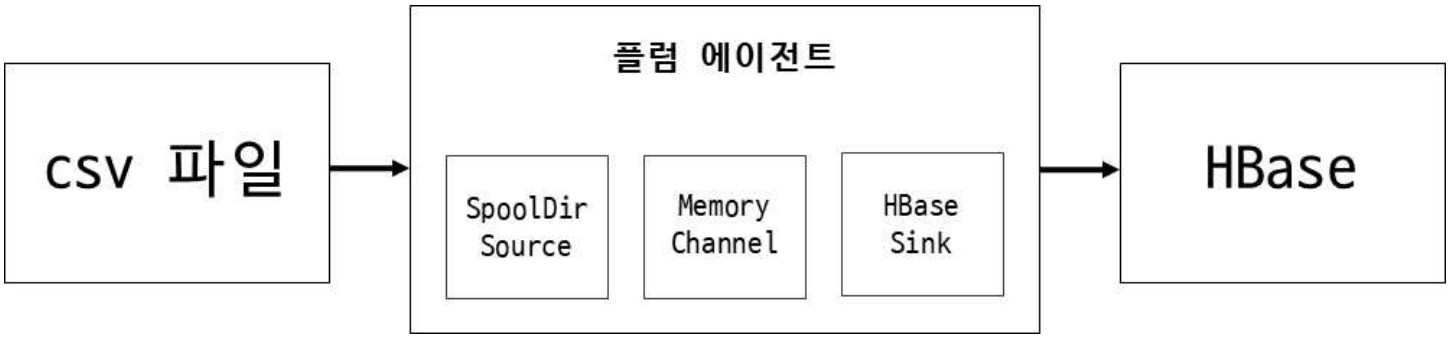
```
$ ./zookeeper-3.4.12/bin/zkServer.sh start
```

- hadoop 시작

```
$ ./hadoop-2.9.1/sbin/start-all.sh
```

○ Agent 이름: Hbase_Agent

○ Hbase_Agent 구성: SpoolDir Source - Memory Channel - HBase Sink



- Hbase Sink

```
Hbase_Agent.sinks.HbaseSink.type = hbase
Hbase_Agent.sinks.HbaseSink.table = test_table
Hbase_Agent.sinks.HbaseSink.columnFamily = cf1
Hbase_Agent.sinks.HbaseSink.serializer                                    =
org.keedio.flume.sink.hbase.serializer.CsvSerializer
Hbase_Agent.sinks.HbaseSink.serializer.columns= date,name,message
```

○ type: hbase로 데이터를 적재

○ table: hbase에 적재할 테이블 명

○ columnFamily: hbase 테이블의 컬럼 패밀리

○ serializer: 사용할 serializer 라이브러리

○ serializer.columns: hbase에 넣을 테이블의 컬럼 수식어

- flume 홈디렉터리의 conf 디렉터리로 이동해 hbaseTest.conf 파일 생성

```
Hbase_Agent.sources  = SpoolSource
Hbase_Agent.channels = MemChannel
Hbase_Agent.sinks    = HbaseSink

Hbase_Agent.sources.SpoolSource.type = spooldir
Hbase_Agent.sources.SpoolSource.spoolDir = /home/bigdata/working/jbm-batch-log
Hbase_Agent.sources.SpoolSource.deletePolicy = immediate
Hbase_Agent.sources.SpoolSource.batchSize = 1000

Hbase_Agent.channels.MemChannel.type = memory
Hbase_Agent.channels.MemChannel.capacity = 100000
Hbase_Agent.channels.MemChannel.transactionCapacity  = 10000

Hbase_Agent.sinks.HbaseSink.type = hbase
Hbase_Agent.sinks.HbaseSink.table = test_table
Hbase_Agent.sinks.HbaseSink.columnFamily = cf1
Hbase_Agent.sinks.HbaseSink.serializer                                          =
org.keedio.flume.sink.hbase.serializer.CsvSerializer
Hbase_Agent.sinks.HbaseSink.serializer.columns=date,name,message

Hbase_Agent.sources.SpoolSource.channels = MemChannel
Hbase_Agent.sinks.HbaseSink.channel = MemChannel
```

- Hbase를 시작

```
$ start-hbase.sh
```

```
[bigdata@server01 bin]$ ./start-hbase.sh
starting master, logging to /home/bigdata/hbase/logs/hbase-bigdata-master-server01.out
server03: starting regionserver, logging to /home/bigdata/hbase/bin/../logs/hbase-bigdata-regi
onserver-server03.out
server02: starting regionserver, logging to /home/bigdata/hbase/bin/../logs/hbase-bigdata-regi
onserver-server02.out
server04: starting regionserver, logging to /home/bigdata/hbase/bin/../logs/hbase-bigdata-regi
onserver-server04.out
server01: starting regionserver, logging to /home/bigdata/hbase/bin/../logs/hbase-bigdata-regi
onserver-server01.out
```

- hbase shell을 열어 test_table 만들기

```
> create 'test_table', 'cf1'
```

- flume홈 디렉터리에서 Hbase_Agent 실행

```
$ ./bin/flume-ng agent -c conf -f conf/hbaseTest.conf -n Hbase_Agent
-Dflume.root.logger=INFO,console
```

```
2018-06-21 14:34:19,399 (lifecycleSupervisor-1-1-SendThread(server01:2181)) [INFO - org.apache.zookee
per.ClientCnxn$SendThread.onConnected(ClientCnxn.java:1235)] Session establishment complete on server
 server01/192.168.56.101:2181, sessionid = 0x1000015ee420012, negotiated timeout = 90000
2018-06-21 14:34:22,041 (lifecycleSupervisor-1-1) [INFO - org.apache.flume.instrumentation.MonitoredC
ounterGroup.register(MonitoredCounterGroup.java:119)] Monitored counter group for type: SINK, name: H
baseSink: Successfully registered new MBean.
2018-06-21 14:34:22,041 (lifecycleSupervisor-1-1) [INFO - org.apache.flume.instrumentation.MonitoredC
ounterGroup.start(MonitoredCounterGroup.java:95)] Component type: SINK, name: HbaseSink started
```

- /home/bigdata/working 에 test2.csv 생성

```
$ cd /home/bigdata/working
```

```
$ vi test2.csv
```

```
20180621,suzy,hello
20180621,iu,hi
20180621,henry,byebye
```

- /home/bigdata/working/jbm-batch-log에 test2.csv를 업로드

```
$  cp test2.csv jbm-batch-log/
```

- Flume이 데이터를 수집

```
2018-06-21 14:57:31,971 (pool-3-thread-1) [INFO - org.apache.flume.client.avro.ReliableSpoolingFileEv
entReader.deleteCurrentFile(ReliableSpoolingFileEventReader.java:492)] Preparing to delete file /home
/bigdata/working/jbm-batch-log/test2.csv
```

- hbase shell에서 test_table 확인해서 적재가 성공적으로 완료되었는 지 확인

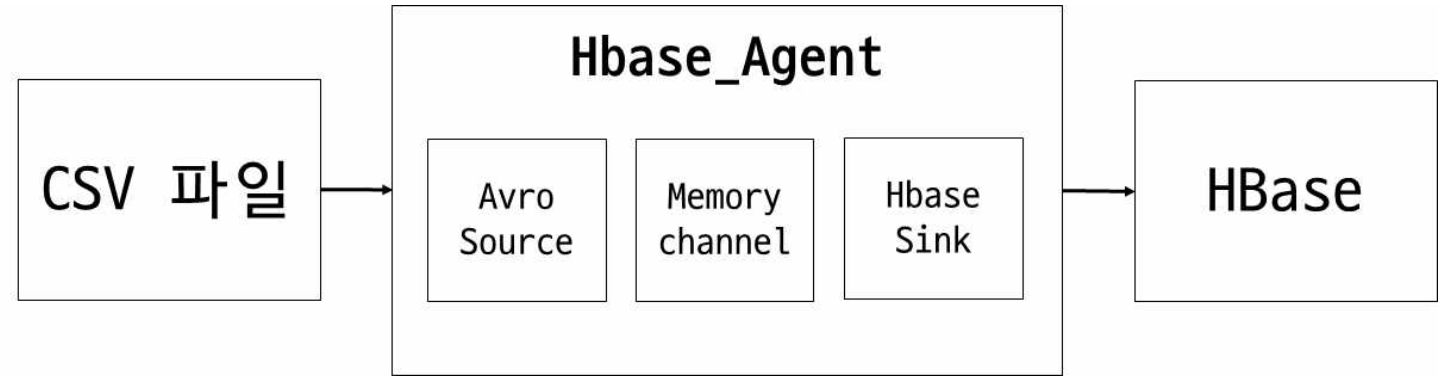
```
$ hbase shell
```

```
>  scan 'test_table'
```

```
hbase(main):002:0> scan 'test_table'
ROW                                   COLUMN+CELL
2fa2d1a7-6b99-4c2f-a638-aaf6b column=cf1:date, timestamp=1529560657855, value=20180621
0e9de01
2fa2d1a7-6b99-4c2f-a638-aaf6b column=cf1:message, timestamp=1529560657855, value=hello
0e9de01
2fa2d1a7-6b99-4c2f-a638-aaf6b column=cf1:name, timestamp=1529560657855, value=suzy
0e9de01
b0d26b74-c22a-4a4d-923f-dd930 column=cf1:date, timestamp=1529560657855, value=20180621
3770754
b0d26b74-c22a-4a4d-923f-dd930 column=cf1:message, timestamp=1529560657855, value=byebye
3770754
b0d26b74-c22a-4a4d-923f-dd930 column=cf1:name, timestamp=1529560657855, value=henry
3770754
d5a8761b-3d23-43a8-b886-82596 column=cf1:date, timestamp=1529560657855, value=20180621
7032b45
d5a8761b-3d23-43a8-b886-82596 column=cf1:message, timestamp=1529560657855, value=hi
7032b45
d5a8761b-3d23-43a8-b886-82596 column=cf1:name, timestamp=1529560657855, value=iu
7032b45
```

○ Agent 이름: Hbase_Agent

○ Hbase_Agent 구성: Avro Source - Memory Channel - HBase Sink



- Flume홈디렉터리의 conf디렉터리에 AvroHbase.conf 생성

```
$ cd /home/bigdata/apache-flume-1.8.0-bin/conf
```

```
$ vi AvroHbase.conf
```

```
Hbase_Agent.sources  = AvroSource
Hbase_Agent.channels = MemChannel
Hbase_Agent.sinks    = HbaseSink

Hbase_Agent.sources.AvroSource.type = avro
Hbase_Agent.sources.AvroSource.bind = 0.0.0.0
Hbase_Agent.sources.AvroSource.port = 65111

Hbase_Agent.channels.MemChannel.type = memory
Hbase_Agent.channels.MemChannel.capacity = 100000
Hbase_Agent.channels.MemChannel.transactionCapacity  = 10000

Hbase_Agent.sinks.HbaseSink.type = hbase
Hbase_Agent.sinks.HbaseSink.table = test_table
Hbase_Agent.sinks.HbaseSink.columnFamily = cf1
Hbase_Agent.sinks.HbaseSink.serializer = org.keedio.flume.sink.hbase.serializer.
Hbase_Agent.sinks.HbaseSink.serializer.columns=date,name,message

Hbase_Agent.sources.AvroSource.channels = MemChannel
Hbase_Agent.sinks.HbaseSink.channel = MemChannel
```

- ConsCompFlumeClient.java

```
package com.sapient.flumeclient;

import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.Reader;
import java.nio.charset.Charset;
import java.util.Iterator;
import java.util.NoSuchElementException;

import org.apache.commons.csv.CSVFormat;
import org.apache.commons.csv.CSVRecord;
import org.apache.flume.Event;
import org.apache.flume.EventDeliveryException;
import org.apache.flume.api.RpcClient;
import org.apache.flume.api.RpcClientFactory;
import org.apache.flume.event.EventBuilder;

public class ConsCompFlumeClient implements Runnable {

	private RpcClient client;
	private String hostname;
	private int port;

	private String[] names = {"iu","suzy","henry","우성","예진","pilgu","동일"};

	public void init(String hostname, int port) {
		// Setup the RPC connection
		this.hostname = hostname;
		this.port = port;
		this.client = RpcClientFactory.getDefaultInstance(hostname, port);

	}
	public void sendDataToFlume() {

	}

	public void cleanUp() {
		client.close();
```

```
	}

	public void run() {
		while (true) {

			try {
				Thread.sleep(10);

				for (int i = 1; i < names.length; i++) {
					Event event =
EventBuilder.withBody("20180621,"+names[i]+",test", Charset.forName("UTF-8"));
					client.append(event);
				}
				System.out.println("test");
				client.close();
			} catch (NoSuchElementException e) {
				System.out.println(e.getMessage());
			} catch (Exception e) {

			}

			client.close();
			client = null;
			client = RpcClientFactory.getDefaultInstance(hostname, port);

		}
	}

}
```

- Flume 홈디렉터리에서 Hbase_Agent 실행

```
$ cd /home/bigdata/apache-flume-1.8.0-bin
```

```
 $  ./bin/flume-ng  agent  -c  conf  -f  conf/hbaseTest.conf  -n  Hbase_Agent
-Dflume.root.logger=INFO,console
```

- ConsCompFlumeClient.java 실행

- 총 적재된 로우(row) 개수 확인

```
> count 'test_table'
```

```
Current count: 121000, row: f8dbf037-089c-43cc-be9d-2e04c3cd07f4
Current count: 122000, row: fae230ea-80d5-428e-9b90-ddc45f05a454
Current count: 123000, row: fced910f-84d9-40ba-a51f-c11898cc2bb1
Current count: 124000, row: fed4564c-9506-4f74-a670-3949bc375ea6
124551 row(s) in 11.3930 seconds

=> 124551
```

- 원하는 컬럼을 5개만 출력(대소문자 구분해야함)

```
> scan 'test_table', {COLUMNS => ['cf1:name','cf1:message'], LIMIT => 5}
```

```
ROW                           COLUMN+CELL
 0000909a-acc2-4da2-b57b-d0 column=cf1:message, timestamp=1529617788086, value=test
 49917be8f5
 0000909a-acc2-4da2-b57b-d0 column=cf1:name, timestamp=1529617788086, value=pilgu
 49917be8f5
 0000bb4d-45b3-42e6-bbd9-9d column=cf1:message, timestamp=1529617780515, value=test
 a5fa982150
 0000bb4d-45b3-42e6-bbd9-9d column=cf1:name, timestamp=1529617780515, value=suzy
 a5fa982150
 00010c7f-66f7-405d-ae10-04 column=cf1:message, timestamp=1529617703785, value=test
 cf2f9fed24
 00010c7f-66f7-405d-ae10-04 column=cf1:name, timestamp=1529617703785, value=pilgu
 cf2f9fed24
 00023e9a-a7ee-48b0-863e-ba column=cf1:message, timestamp=1529617682258, value=test
 dc6c802379
 00023e9a-a7ee-48b0-863e-ba column=cf1:name, timestamp=1529617682258, value=pilgu
 dc6c802379
 000247d2-4337-48e1-8bb7-b2 column=cf1:message, timestamp=1529617481091, value=test
 ad6469a327
 000247d2-4337-48e1-8bb7-b2 column=cf1:name, timestamp=1529617481091, value=dabin
 ad6469a327
5 row(s) in 0.0590 seconds
```

- Suzy가 보낸 메세지를 3개만 출력

```
> scan 'test_table', {COLUMNS => ['cf1:name','cf1:message'], FILTER =>
"SingleColumnValueFilter ('cf1', 'name',=,'regexstring:suzy') AND  PageFilter(3)"}
```

```
ROW                              COLUMN+CELL
 0000bb4d-45b3-42e6-bbd9-9d column=cf1:message, timestamp=1529617780515, value=test
 a5fa982150
 0000bb4d-45b3-42e6-bbd9-9d column=cf1:name, timestamp=1529617780515, value=suzy
 a5fa982150
 000743aa-acd7-4f2e-ba42-0f column=cf1:message, timestamp=1529617836266, value=test
 ce14abd6cc
 000743aa-acd7-4f2e-ba42-0f column=cf1:name, timestamp=1529617836266, value=suzy
 ce14abd6cc
 0009ee6a-5c95-43c9-9311-62 column=cf1:message, timestamp=1529617981996, value=test
 d4f61616d8
 0009ee6a-5c95-43c9-9311-62 column=cf1:name, timestamp=1529617981996, value=suzy
 d4f61616d8
3 row(s) in 0.0690 seconds
```

○ regexstring: 지정된 정규식을 사용하여 지정된 바이트 배열과 비교

- 우성이 보낸 메세지를 3개만 출력

```
> scan 'test_table', {COLUMNS => ['cf1:name:toString','cf1:message:toString'], FILTER
=> "SingleColumnValueFilter ('cf1', 'name',=,'regexstring:우성') AND  PageFilter(3)"}
```

```
ROW                                COLUMN+CELL
 00945de2-cf27-4558-a318-f1bc19 column=cf1:message, timestamp=1529628891234, value=test
 fc1fa4
 00945de2-cf27-4558-a318-f1bc19 column=cf1:name, timestamp=1529628891234, value=우성
 fc1fa4
 01a2d81b-c3ae-4088-af99-557a1e column=cf1:message, timestamp=1529628884923, value=test
 6065b3
 01a2d81b-c3ae-4088-af99-557a1e column=cf1:name, timestamp=1529628884923, value=우성
 6065b3
 01da6647-de81-4eb0-9cfe-3b8b54 column=cf1:message, timestamp=1529628884569, value=test
 fcbf35
 01da6647-de81-4eb0-9cfe-3b8b54 column=cf1:name, timestamp=1529628884569, value=우성
 fcbf35
3 row(s) in 0.0360 seconds
```

○ 컬럼:toString으로 값을 String 형태로 확인 가능

- 특정 로우키(Row Key)인 ffff7607를 포함하고 있는 로우(Row) 출력

```
> scan 'test_table', {FILTER => "RowFilter(=,'regexstring:ffff7607')"}
```

```
ROW                           COLUMN+CELL
 ffff7607-ed04-4d12-86a8-c9 column=cf1:date, timestamp=1529617705817, value=20180621
 3b8601e672
 ffff7607-ed04-4d12-86a8-c9 column=cf1:message, timestamp=1529617705817, value=test
 3b8601e672
 ffff7607-ed04-4d12-86a8-c9 column=cf1:name, timestamp=1529617705817, value=suzy
 3b8601e672
1 row(s) in 0.2340 seconds
```

- hen으로 이름이 시작되는 사람 찾기

```
> scan 'test_table', {LIMIT => 10,
FILTER=>"SingleColumnValueFilter('cf1','name',=,'regexstring:hen*')"}
```

```
ROW                           COLUMN+CELL
 4f8c1066-ad00-460c-82f6-4a column=cf1:date, timestamp=1529561186675, value=20180621
 05dbc614e5
 4f8c1066-ad00-460c-82f6-4a column=cf1:message, timestamp=1529561186675, value=byebye
 05dbc614e5
 4f8c1066-ad00-460c-82f6-4a column=cf1:name, timestamp=1529561186675, value=henry
 05dbc614e5
1 row(s) in 1.2100 seconds
```

- 201806으로 시작하는 값의 등록 역순으로 100개만 조회

```
> scan 'test_table', {REVERSED=>true, LIMIT=>100,
FILTER=>"SingleColumnValueFilter('cf1','date',=,'regexstring:201806*')"}
```

```
ROW                          COLUMN+CELL
 ffffa08e-2ec4-4a90-87b6-9f column=cf1:date, timestamp=1529617857868, value=20180621
 c32ed568e2
 ffffa08e-2ec4-4a90-87b6-9f column=cf1:message, timestamp=1529617857868, value=test
 c32ed568e2
 ffffa08e-2ec4-4a90-87b6-9f column=cf1:name, timestamp=1529617857868, value=pilgu
 c32ed568e2
 ffff7607-ed04-4d12-86a8-c9 column=cf1:date, timestamp=1529617705817, value=20180621
 3b8601e672
 ffff7607-ed04-4d12-86a8-c9 column=cf1:message, timestamp=1529617705817, value=test
 3b8601e672
 ffff7607-ed04-4d12-86a8-c9 column=cf1:name, timestamp=1529617705817, value=suzy
 3b8601e672
 fffe6793-d478-41c1-971e-9f column=cf1:date, timestamp=1529617360311, value=20180621
 16f1c65dea
```

- 로우키에 bbbb를 포함하지 않는 로우(Row)를 3개만 출력

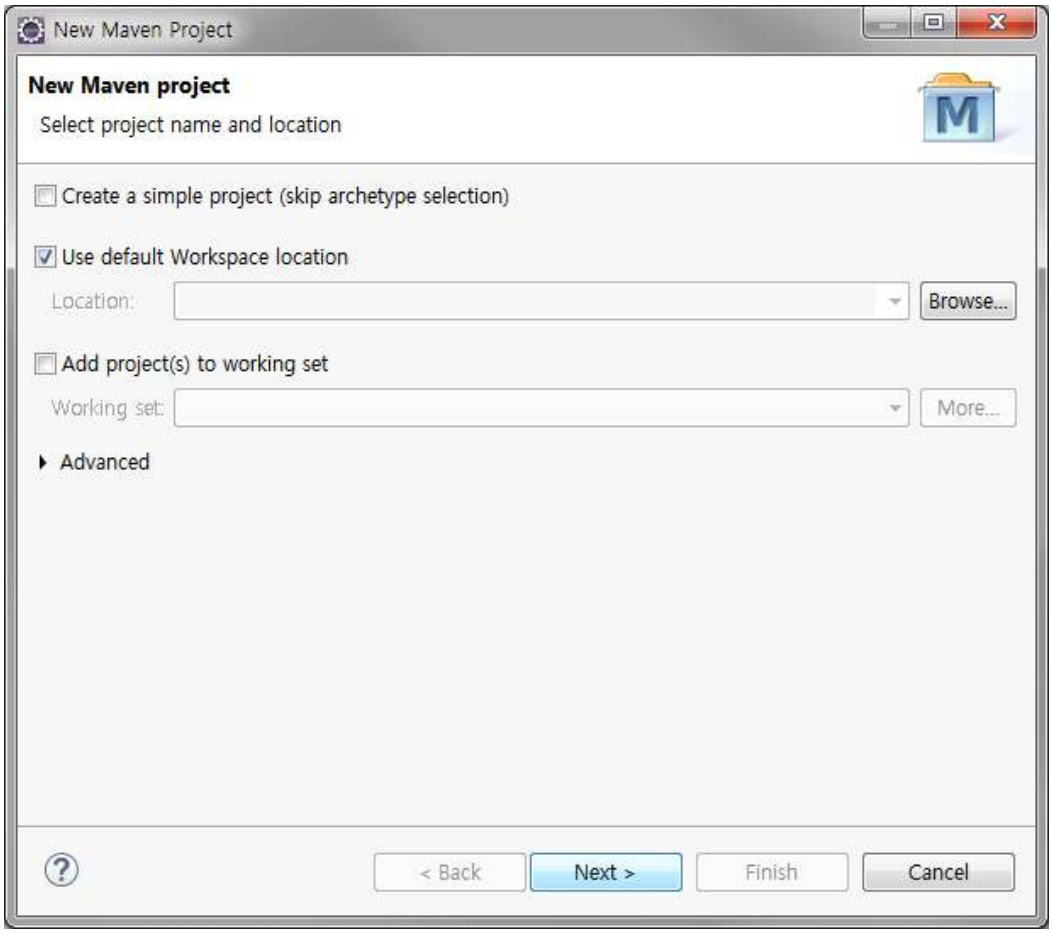
```
> scan 'test_table', {FILTER => "RowFilter(!=,'regexstring:bbbb')", LIMIT => 3}
```

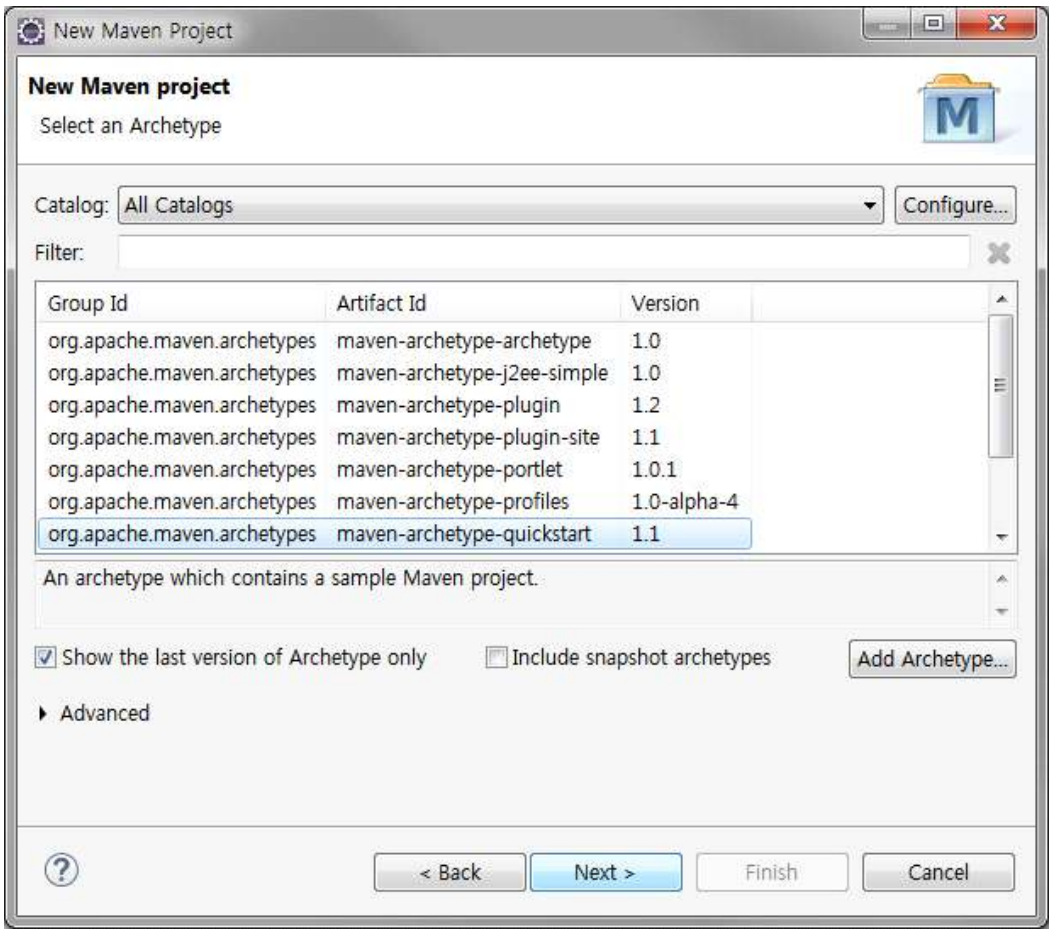
```
ROW                          COLUMN+CELL
 0000909a-acc2-4da2-b57b-d0 column=cf1:date, timestamp=1529617788086, value=20180621
 49917be8f5
 0000909a-acc2-4da2-b57b-d0 column=cf1:message, timestamp=1529617788086, value=test
 49917be8f5
 0000909a-acc2-4da2-b57b-d0 column=cf1:name, timestamp=1529617788086, value=pilgu
 49917be8f5
 0000bb4d-45b3-42e6-bbd9-9d column=cf1:date, timestamp=1529617780515, value=20180621
 a5fa982150
 0000bb4d-45b3-42e6-bbd9-9d column=cf1:message, timestamp=1529617780515, value=test
 a5fa982150
 0000bb4d-45b3-42e6-bbd9-9d column=cf1:name, timestamp=1529617780515, value=suzy
 a5fa982150
 00010c7f-66f7-405d-ae10-04 column=cf1:date, timestamp=1529617703785, value=20180621
 cf2f9fed24
 00010c7f-66f7-405d-ae10-04 column=cf1:message, timestamp=1529617703785, value=test
 cf2f9fed24
 00010c7f-66f7-405d-ae10-04 column=cf1:name, timestamp=1529617703785, value=pilgu
 cf2f9fed24
3 row(s) in 0.0360 seconds
```

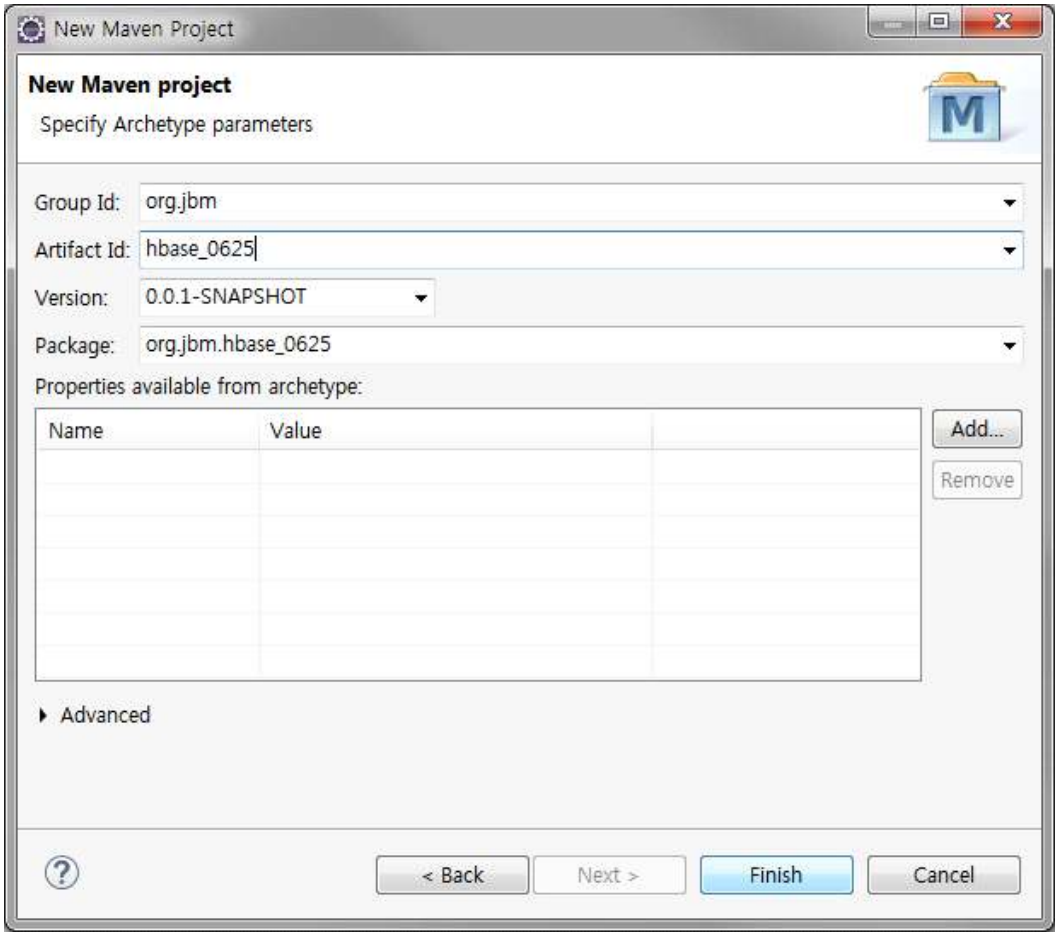
○ regexstring 검색은 =, != 만 지원

■ HBase 프로그래밍

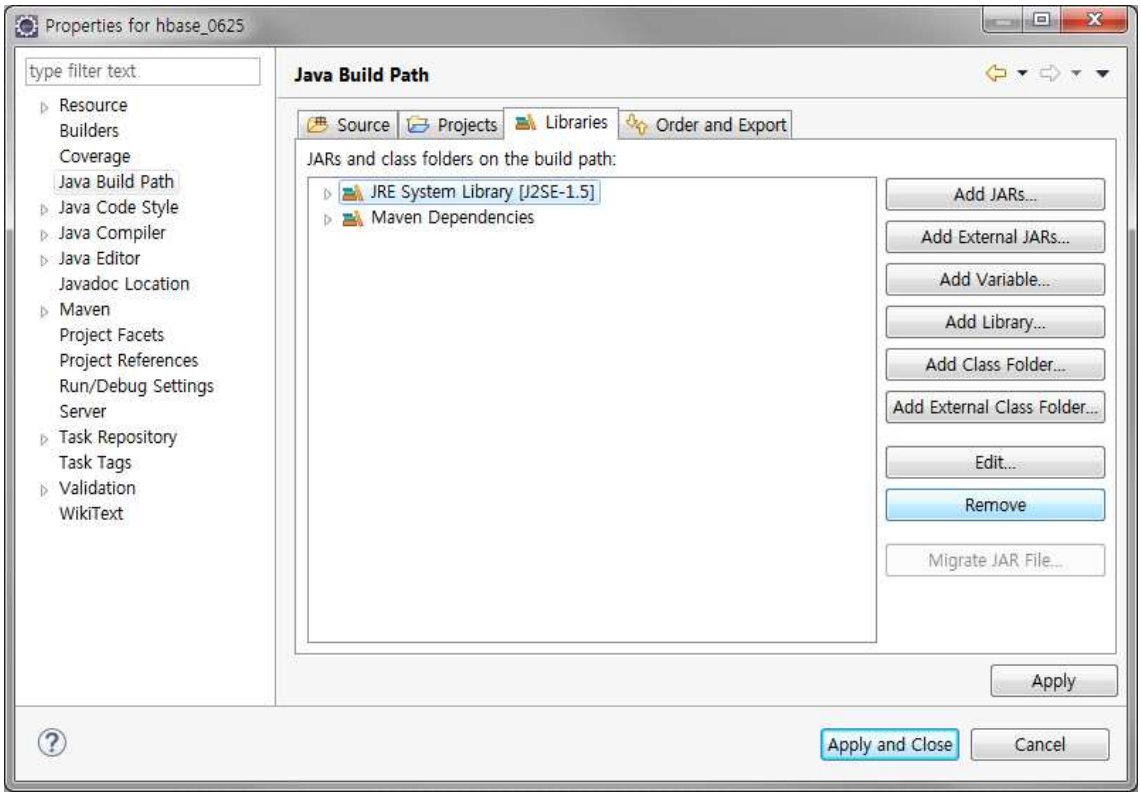
1) maven 프로젝트 생성

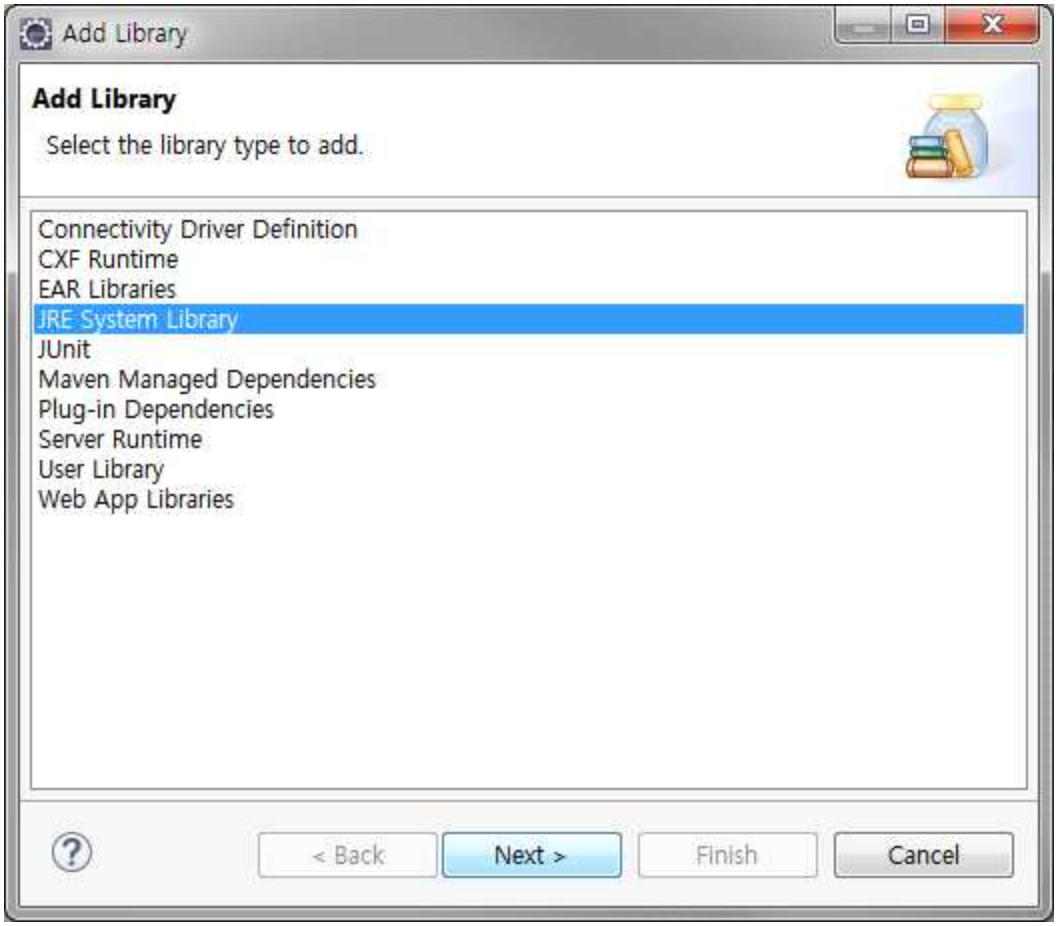


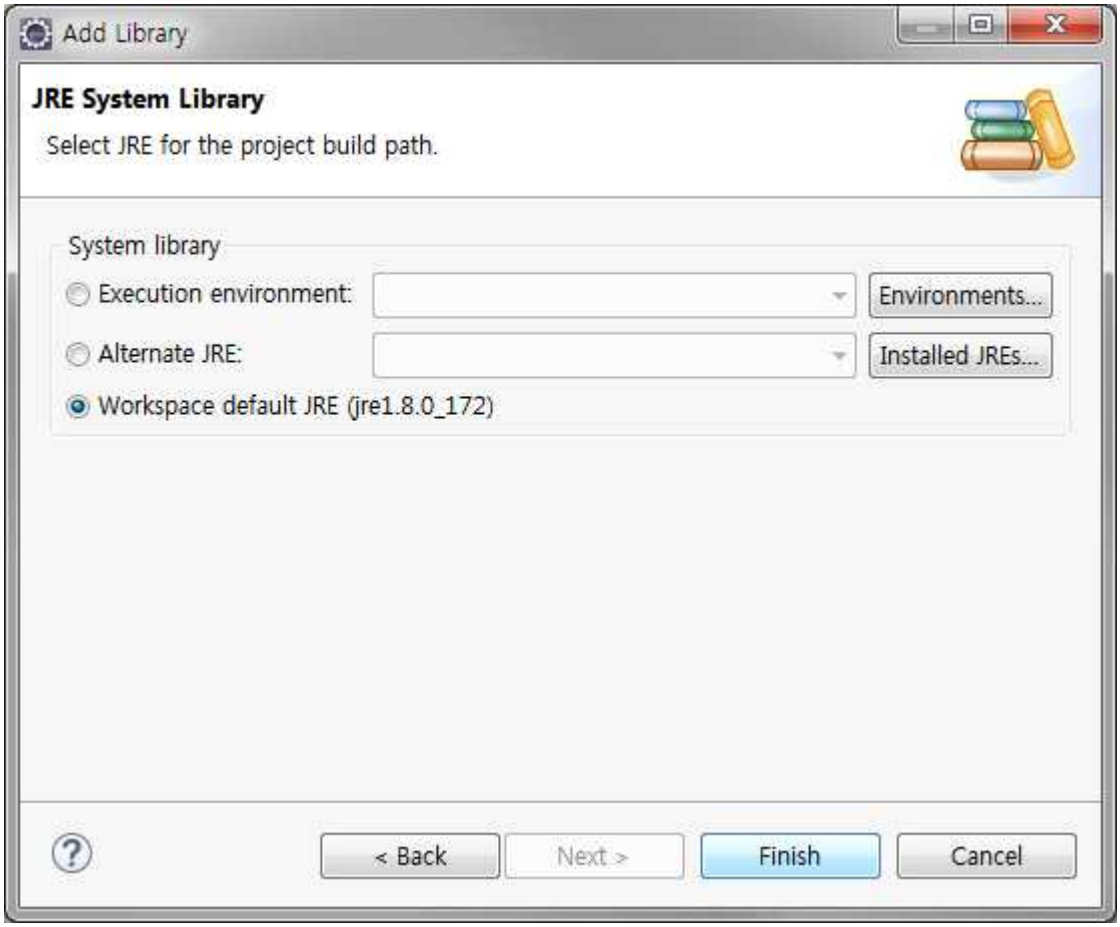


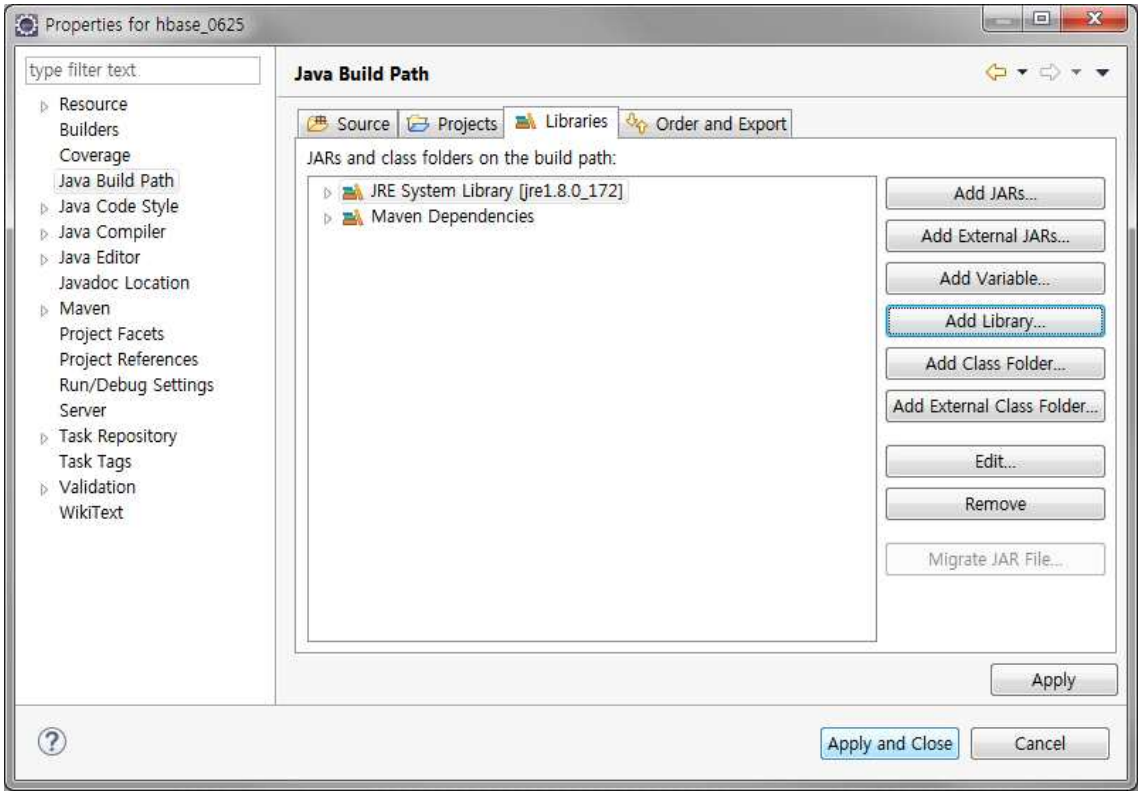


2) JRE 변경(기본 1.5에서 1.8로)




Add Library
Add Library
Select the library type to add.
Connectivity Driver Definition
CXF Runtime
EAR Libraries
JRE System Library
JUnit
Maven Managed Dependencies
Plug-in Dependencies
Server Runtime
User Library
Web App Libraries
< Back
Next >
Finish
Cancel


Add Library
JRE System Library
Select JRE for the project build path.
System library
Execution environment:
Environments...
Alternate JRE:
Installed JREs...
Workspace default JRE (jre1.8.0_172)
< Back
Next >
Finish
Cancel



3) pom.xml에 hbase-client 추가

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
	xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
	xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
	<modelVersion>4.0.0</modelVersion>

	<groupId>org.jbm</groupId>
	<artifactId>hbase_0625</artifactId>
	<version>0.0.1-SNAPSHOT</version>
	<packaging>jar</packaging>

	<name>hbase_0625</name>
	<url>http://maven.apache.org</url>

	<properties>
		<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
	</properties>

	<profiles>
		<profile>
			<id>windows-profile</id>
			<activation>
				<activeByDefault>true</activeByDefault>
				<file>
					<exists>${JAVA_HOME}/lib/tools.jar</exists>
```

```
                        </file>
                    </activation>
                    <properties>
                        <toolsjar>${JAVA_HOME}/lib/tools.jar</toolsjar>
                    </properties>
                </profile>
        </profiles>

        <dependencies>

                <dependency>
                        <groupId>org.apache.hbase</groupId>
                        <artifactId>hbase-client</artifactId>
                        <version>1.2.6</version>
                </dependency>

                <dependency>
                        <groupId>jdk.tools</groupId>
                        <artifactId>jdk.tools</artifactId>
                        <version>jdk1.8.0</version>
                        <scope>system</scope>
                        <systemPath>${toolsjar}</systemPath>
                </dependency>

        </dependencies>
</project>
```

4) HBaseAPIApp 만들기 – SELECT(조회)

```
package org.jbm.hbase_0625;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.client.ResultScanner;
import org.apache.hadoop.hbase.client.Scan;
import org.apache.hadoop.hbase.client.Table;

public class HBaseApp {

        public static void main(String[] args) throws Exception {

                Configuration config = HBaseConfiguration.create();
```

```
                config.clear();

                config.set("hbase.master", "192.168.56.101");
                config.set("hbase.zookeeper.quorum", "192.168.56.101");
                config.set("hbase.zookeeper.property.clientPort", "2181");

                Connection connection = ConnectionFactory.createConnection(config);

                TableName tname = TableName.valueOf("chattings");

                Table table = connection.getTable(tname);

                // 테이블 데이터 조회
                Scan scan = new Scan();

                ResultScanner rs = table.getScanner(scan);

                for (Result result : rs) {

                        System.out.println(result);

                }//for end

        }// main() end

}// HBaseApp end
```

5) HBaseAPIApp2 만들기 (입력 : insert)

```
package org.jbm.hbase_0625;

import java.util.UUID;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.TableName;
import org.apache.hadoop.hbase.client.Connection;
import org.apache.hadoop.hbase.client.ConnectionFactory;
import org.apache.hadoop.hbase.client.Put;
import org.apache.hadoop.hbase.client.Table;
import org.apache.hadoop.hbase.util.Bytes;

public class HBaseApp2 {

        public static void main(String[] args) throws Exception{
```

```
                //row키를 랜덤하게 설정
                byte[] row1 = Bytes.toBytes(UUID.randomUUID().toString());

                byte[] family1 = Bytes.toBytes("message");
                byte[] family2 = Bytes.toBytes("location");

                byte[] qualifier1 = Bytes.toBytes("name");
                byte[] qualifier2 = Bytes.toBytes("message");
                byte[] qualifier3 = Bytes.toBytes("country");
                byte[] qualifier4 = Bytes.toBytes("city");

                Configuration config = HBaseConfiguration.create();

        config.clear();

        config.set("hbase.master", "192.168.56.101");
        config.set("hbase.zookeeper.quorum", "192.168.56.101");
        config.set("hbase.zookeeper.property.clientPort", "2181");

        Connection connection =  ConnectionFactory.createConnection(config);
        TableName tname = TableName.valueOf("table4");

        Table table = connection.getTable(tname);

        Put p = new Put(row1);


      p.addColumn(family1, qualifier1, "pilgu".getBytes());
      p.addColumn(family1, qualifier2, "안녕하세요? 반갑습니다~".getBytes());
      p.addColumn(family2, qualifier3, "korea".getBytes());
      p.addColumn(family2, qualifier4, "seoul".getBytes());

      table.put(p);

      System.out.println("성공!");


        }//main() end

}//HBaseApp2 end
```