Stream Reasoning
For Linked Data
M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, and E. Della Valle
http://streamreasoning.org/events/sr4ld2014

Morph-streams: Hands on Session
Jean-Paul Calbimonte
jean-paul.calbimonte@epfl.ch
lsir.epfl.ch
Share, Remix, Reuse — Legally

- This work is licensed under the Creative Commons Attribution 3.0 Unported License.

- **Your are free:**
  
  - **to Share** — to copy, distribute and transmit the work
  
  - **to Remix** — to adapt the work

- **Under the following conditions**
  
  - **Attribution** — You must attribute the work by inserting
    - "[source http://streamreasoning.org/rsp2014]" at the end of each reused slide
    - a credits slide stating
      - These slides are partially based on “Streaming Reasoning for Linked Data 2013” by M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, E. Della Valle http://streamreasoning.org/rsp2014

- To view a copy of this license, visit
  [http://creativecommons.org/licenses/by/3.0/](http://creativecommons.org/licenses/by/3.0/)
A bit of Morph-streams

- What we will cover:
  - SPARQLstream queries
  - Register queries
  - Pull data
  - Push data

- Morph-web: a demo web application for Morph-streams
  - https://github.com/jpcik/morph-web
  - Install it yourself (follow the instructions in github)
Hands-on instructions

- The instructions are on the github wiki:

- We’ll be using this server for the hands-on:
  - http://streams.linkeddata.es
Use cases in the Demo

- You can choose one of the use cases in the Demo home:

  Choose a demo System:
  - Social Sensor Demo (running Esper)
  - EMT Bus stations Madrid (running GSN)
  - Citybikes urban Bike sensors (running GSN)
  - Swiss Experiment environmental data (running GSN)
  - HL7 synthetic patient data (running Esper)
Social Sensor Use Case

- In short: People detected in rooms
- Use Esper as datasource

### detections {roomid:string, person:string, time:string}

For example, this stream may contain tuples as the following:

```
r1,alice,2013-10-10T10:00
```

But of course we want to query this data through an ontology...
Query using an Ontology

Let's use this ontology:

Oversimplified ontology: an observation encapsulates something that a sensor has observed.

**who** was observed (a person), and **where** (in a room).
Continuous query

• Go to MORPH_HOST/query/social.
• Write a query or choose one

• e.g. all observations when carl was detected in the last 30 seconds:

```
PREFIX sr4ld: <http://streamreasoning.org/ontologies/social#>
PREFIX pers: <http://streamreasoning.org/data/person/id/>
SELECT ?obs
FROM NAMED STREAM <http://streamreasoning.org/data/social.srdf>
[NOW - 30 S]
WHERE {
}
```

• Only registered the query. to see some data pull results.
Pull Data

- The query has been given an identifier
- Can be used to retrieve results by pulling.

![Morph Streams](morph_streams.png)

- You can also remove the query when you no longer need it.
Listen to a query

- Receive results as soon as they are available
- Using a WebSocket.
- WebSockets implement full-duplex communication via TCP, and are supported by most browsers.

ws://linkeddata2.dia.fi.upm.es:9000/push?
query=PREFIX%20sr4l....
If issues with WebSocket

- **Alternative server:**
  - http://linkeddata2.dia.fi.upm.es:9000
Changing the mappings

For example you can change the URI template for a Person, instead of this predicate map:

```sparql
]];
```

You can define the following:

```sparql
]];
```
Underlying queries

- Underlying queries checkbox
- To see what is being sent to the DSMS or CEP
EMT Bus stops Madrid

- Using GSN
- Instantaneous one-off queries
  - get all bus stop observations in the last 5 mins:

```sparql
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX qudt: <http://data.nasa.gov/qudt/owl/qudt#>
PREFIX emt: <http://emt.linkeddata.es/data#>
SELECT ?timeto ?obs ?av
FROM NAMED STREAM <http://emt.linkeddata.es/data#busstops.srdf> [NOW - 300 S]
WHERE {
  ?obs a emt:BusObservation.
  ?output emt:timeToBusValue ?av.
}
```
One-off query results

- Fire and forget
Morph-streams as REST service

- MORPH_HOST/emt/sparqlstream?query=ENCODERQUERY

- ‘ENCODERQUERY’ is the SPARQLStream encoded for a URL. E.g.:

  http://linkeddata2.dia.fi.upm.es:9000/emt/sparqlstream?query=PREFIX%20ssn%3A%20%3Chttp%3A//purl.oclc.org/NET/ssnx/ssn%23%3E%0APREFIX%20qudt%3A%20%3Chttp%3A//data.nasa.gov/qudt/owl/qudt%23%3E%0APREFIX%20emt%3A%20%3Chttp%3A//emt.linkeddata.es/data%23%3E%0ASELECT%20%3Ftimeto%20%3Fobs%20%3Fav%20%0AFROM%20NAMED%20STREAM%3Chttp%3A//emt.linkeddata.es/data%20%3Bsbusstops.srdf%3E%20%5BNOW%20-%20300%20S%5D%0A%20%7B%0A%20%20%3Fobs%20a%20emt%3ABusObservation.%0A%20%20%3Fobs%20ssn%3AobservationResult%20%3Foutput.%0A%20%20%20%3Foutput%20emt%3AtimeToBusValue%20%3Fav.%0A%20%20%20%3Fav%20qudt%3AnumericValue%20%3Ftimeto.%0A%7D

A bit ugly but it’s a kind of SPARQLStream endpoint
Getting the results

```json
{
    "head": {
        "vars": [ "timeto" , "obs" , "av" ]
    },
    "results": {
        "bindings": [
            {
                "timeto": { "datatype": "http://www.w3.org/2001/XMLSchema#string" , "type": "typed-literal" , "value": "999999" } ,
                "obs": { "type": "uri" , "value": "http://transporte.linkeddata.es/emt/busstop/id/44/busline/147/observation/20/10/2013%2010:35:38%20%2B0200" } ,
                "av": { "type": "uri" , "value": "http://transporte.linkeddata.es/emt/busstop/id/44/busline/147/timeToBusValue/20/10/2013%2010:35:38%20%2B0200" }
            } ,
        ]
    }
}
```
More mapping changes

Add a predicate object map

```
rr:predicateObjectMap [  
  rr:predicate sr4ld:when;  
  rr:objectMap [rr:column "time"]];
```
EMT Buses (R2RML graphical view)

BusStop SensingDevice

observedBy

BusLine

busLine

BusObservation

observation Result

BusStop SensorOutput

observation ResultTime

time:Instant

TimeTo BusValue

DistanceTo BusValue
Stream Reasoning
For Linked Data
M. Balduini, J-P Calbimonte, O. Corcho, D. Dell'Aglio, and E. Della Valle
http://streamreasoning.org/events/sr4ld2014

Morph-streams: Hands on Session