Reading Triplets

To query triples from the repository, you can use the tupelo operator TripleMatcher, binding any of subject, predicate or object to the constants in the pattern you wish to match. Following the previous example, to retrieve the temperature for Champaign on September 7th you could use the following code:

```
import edu.uiuc.ncsa.datastream.rdf.TimeAnnotatedResource;
import edu.uiuc.ncsa.datastream.time.TemporalEntity;

import org.tupeloproject.rdf.Resource;
import org.tupeloprohect.rdf.Triple;
import org.tupeloproject.kernel.TripleMatcher;

//...

Resource stream = Resource.uriRef("urn:ChampaignWeather");
Resource pred = Resource.uriRef("urn:hasMaxTemperature");

Date d1 = new Date();
d1.set .... // set d1 to 2009-09-07 CST
//create an annotated resource for ChampaignWeather["2009-09-07-06:00"]
Resource subj1 = TimeAnnotatedResource.create(stream, TemporalEntity.getInstant(d1));

TripleMatcher stm = new TripleMatcher();
stm.setSubject(subj1);
stm.setPredicate(pred);

context.perform(stm);

for(Triple triple : stm.getResults())
  processResult(triple.getObject()); //do something with the result
```

The Semantic Data Stream Manager also allows the annotation of resources in ways that essentially create variables, i.e, it allows time annotations that do not represent a concrete, global instant but make a relative reference to a timstamp or set of timestamps. This are annotations such as time ranges, *, FIRST and LAST. You can use these annotations to query RDF data but notice that triples with resources annotated this way cannot be used in a triple writer. For example, to fetch all the triples describing Champaigns, temperature in July 2009:

```
import edu.uiuc.ncsa.datastream.rdf.TimeAnnotatedResource;
import edu.uiuc.ncsa.datastream.time.TemporalAnnotation;

import org.tupeloproject.rdf.Resource;
import org.tupeloprohect.rdf.Triple;
import org.tupeloproject.kernel.TripleMatcher;

//...

Resource stream = Resource.uriRef("urn:ChampaignWeather");
Resource pred = Resource.uriRef("urn:hasMaxTemperature");

Date d1 = new Date();
Date d2 = new Date();
d1.set .... // set d1 to 2009-07-01 CST
d2.set ...  // set d2 to 2009-07-31 CST
//create an annotated resource for ChampaignWeather["2009-07-01-05:00".."2009-07-01-05:00"]
Resource subj1 = TimeAnnotatedResource.create(stream, TemporalAnnotation.getInterval(d1,d2));

TripleMatcher stm = new TripleMatcher();
stm.setSubject(subj1);
stm.setPredicate(pred);

context.perform(stm);

for(Triple triple : stm.getResults())
  processResult(triple.getObject()); //do something with the result
```

Finally, if you need to match a pattern like ?x["2009-09-07-05:00"] <urn:hasMaxTemperature> "77" , where was the temperature 77 on September 7th, you can use the class TimeAnnotatedVariable in aTripleMatcher:

```
Date d1 = new Date();
d1.set ...// set d1 to 2009-09-07 CST
Resource pred = Resource.uriRef("urn:hasMaxTemperature");
TimeAnnotatedVariable subj = new TimeAnnotatedVariable("x",TemporalAnnotation.getInstant(d1));
Resource obj = Resource.literal("77");

TripleMatcher stm = new TripleMatcher();
stm.setPattern(new Pattern(subj,pred,obj));

context.perform(stm);

for(Triple triple : stm.getResult())
  processResult(triple.getSubject()); //do something with the results
```