Application Development

universität innsbruck

National University of Ireland, Galway
Ollscoil na hÉireann, Gaillimh

Universität Stuttgart
Germany

TU WIEN
Technische Universität Wien
Vienna University of Technology

Freie Universität Berlin

ontotext
Semantic Technology Lab

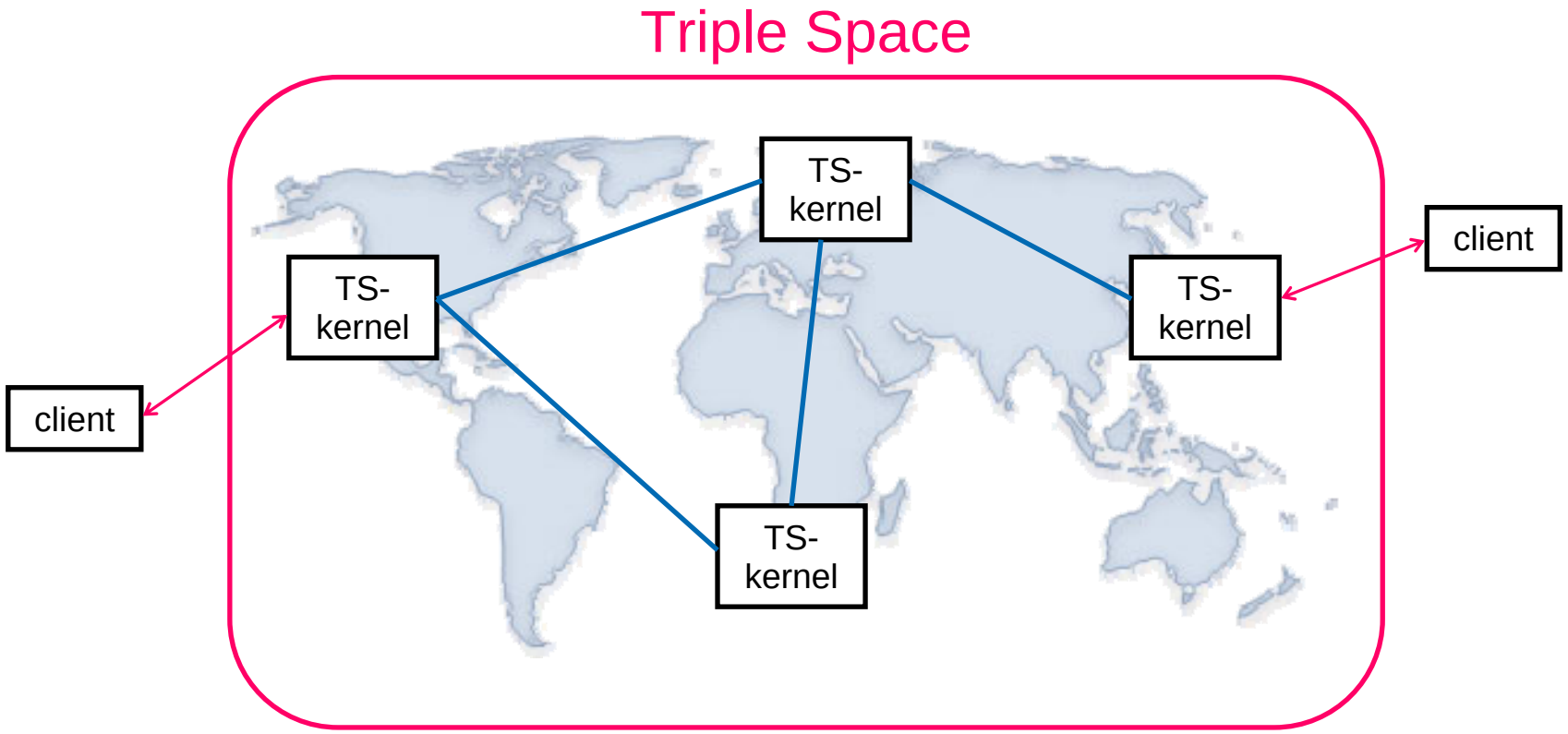
profium

CEFRIEL
FORGING INNOVATION

Telefónica
Telefónica Investigación y Desarrollo

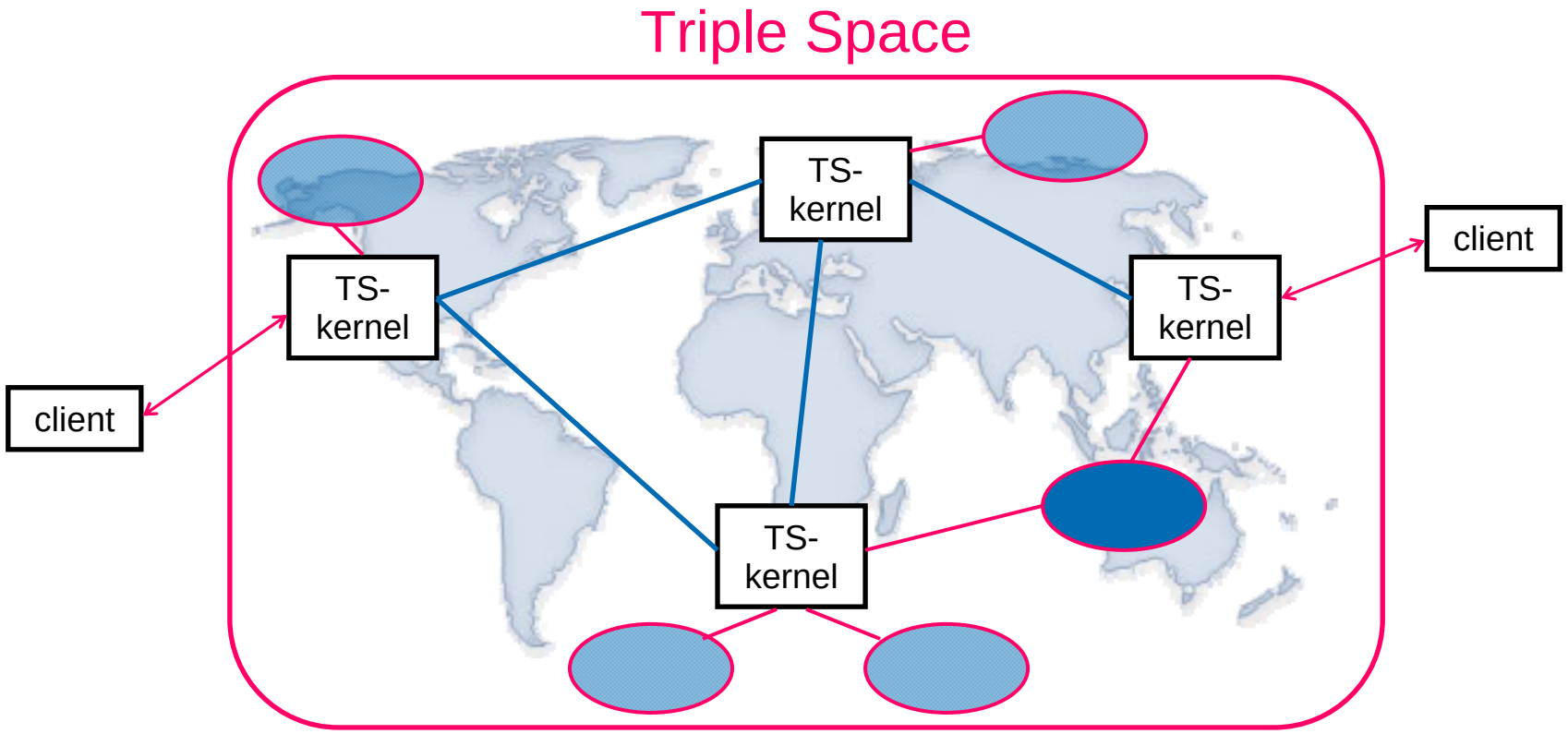
Client Access

- Connect to **one** Triple Space (TS-) kernel
- Read/Write semantic data via Triple Space API
- Interact with other clients



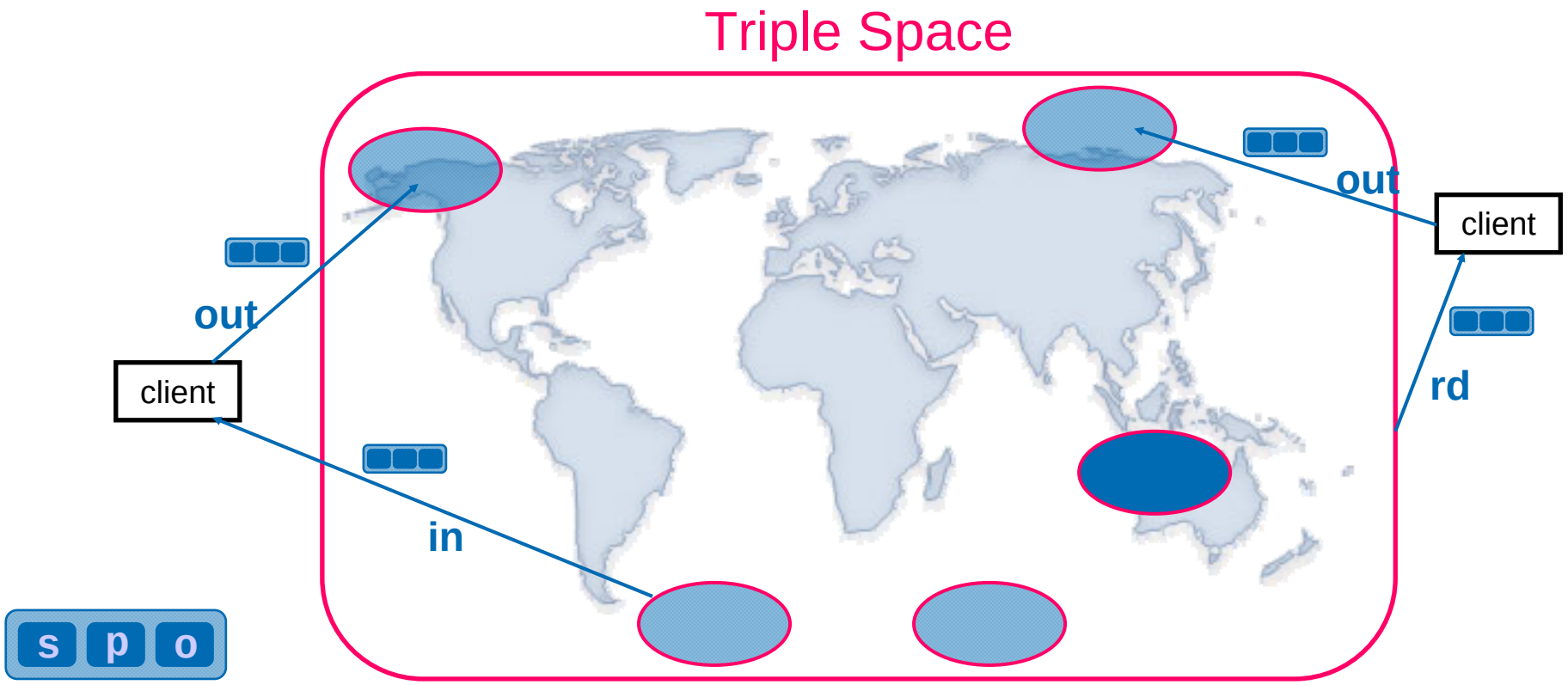
Structure of Triple Space

- Composed of single spaces
  - logical entity, addressed via unique URI
  - stored on one kernel, or distributed onto more kernels



Structure of Triple Space

- Triple Space access operations
  - **out**, **in, rd**: adress particular triplespace („with URI“)
  - **rd** can also adress Triple Space („without URI“)



Terms

- A (tuple-)space is a shared associative memory whose elementary data units are tuples. A tuple is a heterogeneous, ordered collection of typed values
- A triplespace is a tuplespace in which every tuple represents an RDF statement in terms of a triple (subject, predicate, object)
- A triplespace is identified by a unique identifier, i.e., the name of the triplespace
- A triplespace is accessed via the Triple Space (TS) API primitives

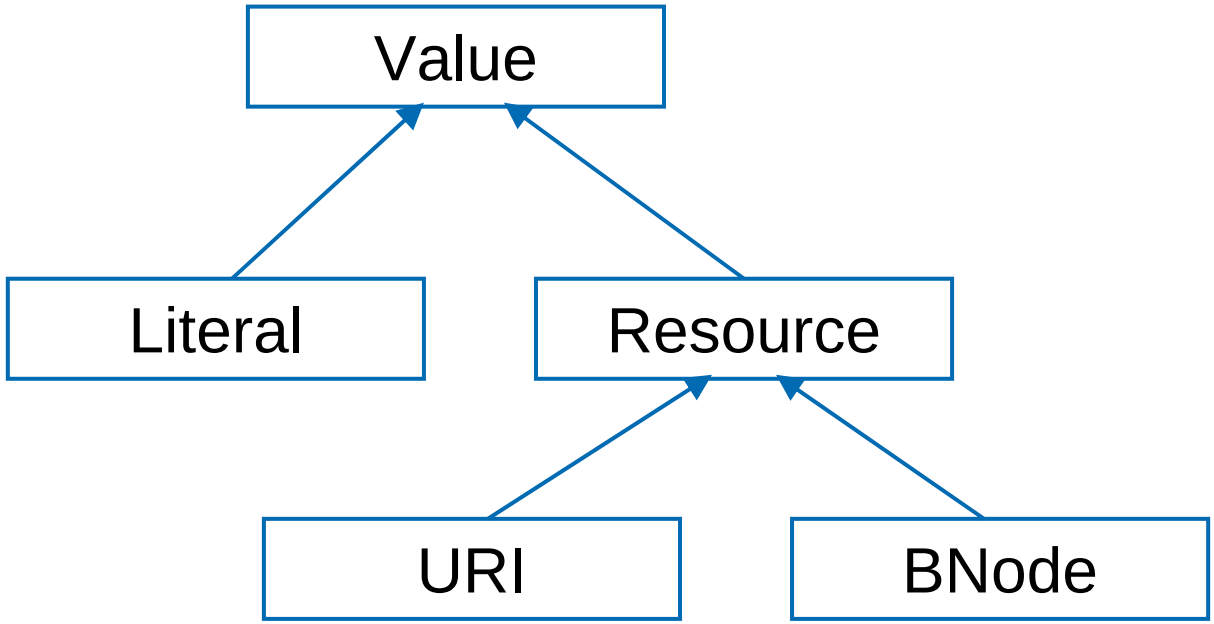
Terms (2)

- A triplespace can be divided into virtual subspaces
- A subspace is a part of a triplespace that complies to the definition of a triplespace and can have subspaces itself
- If a triple is in two spaces, then one of them must be a subspace of the other, i.e., subspaces cannot arbitrarily overlap.
- A space which is not a subspace is termed a root space

Sixth Framework Programme Priority 2
Information Society Technologies (IST)
Specific Targeted Research Project

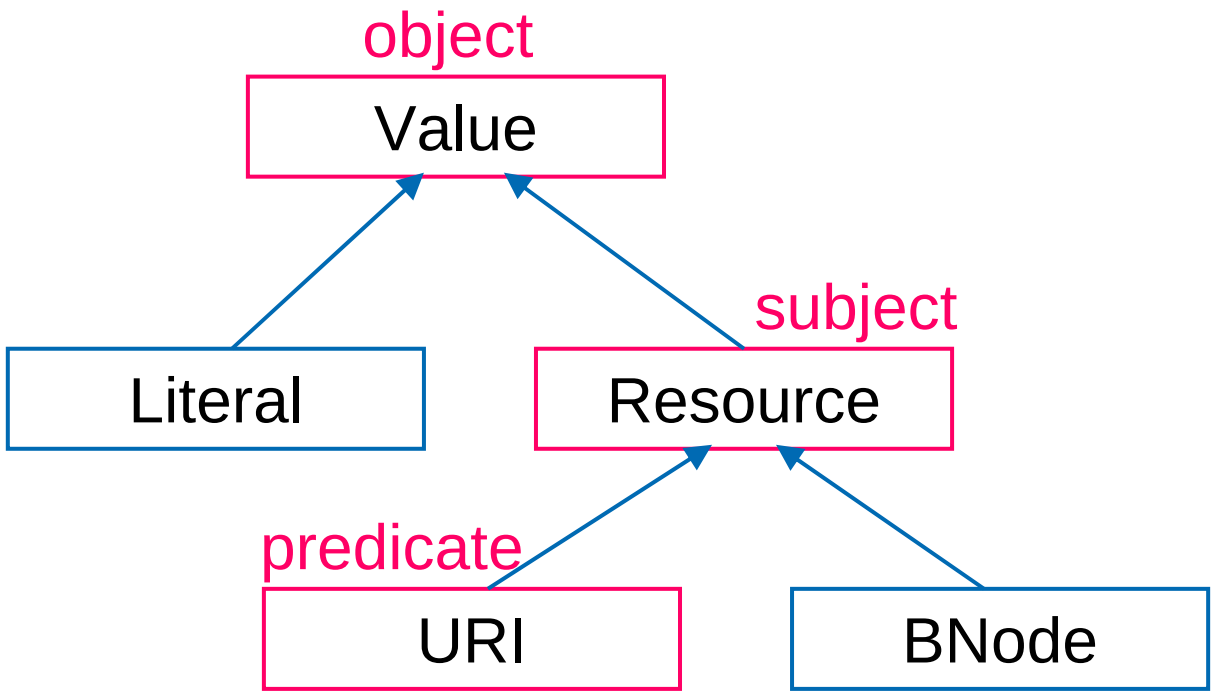
Representation of RDF Triples

- Defined in the OpenRDF Sesame API
- www.openrdf.org/doc/sesame2/2.0.1/apidocs/



Representation of RDF Triples

- Defined in the OpenRDF Sesame API
- www.openrdf.org/doc/sesame2/2.0.1/apidocs/
- RDF triple: org.openrdf.model.Statement



Core TS API

- interface org.tripcom.api.ts.CoreAPI
  - void **out**(Statement **triple**, URI **space**)
    - writes a single triple into the space
    - no guarantee if and when the triple will be available in the space
    - client is immediately free to perform further activities
    - client has to provide a resolvable URI which identifies a space
  - Set<Statement> **rd**(String **query**, URI **space**, int **timeout**)
    - query is a single triple pattern: triple with subject, predicate, or object being variable or exact value (?s, p, o), (?s, ?p, o), ...
    - returns one match of the query
    - return may be set of triples, e.g., Concise Bounded Description
    - no guarantee when the match would be returned
    - timeout gives bound for returning a match, timeout = 0: no timeout
    - if no match has been found within timeout, an empty set is returned
    - allows no conclusion regarding the existence of a match
  - Set<Statement> **rd**(String **query**, int **timeout**)
    - as above, without space URL
    - system can select a match from anywhere in Triple Space

Extended TS API

- interface org.tripcom.api.ts.ExtendedAPI
  - void **out**(Set<Statement> **triples**, URI **space**)
    - like CoreAPI.out, but writes set of triples
    - *deprecated:*
      - *time value can be specified which defines the minimal lifetime of the triples emitted to the space*
      - *after the lifetime, there is no guarantee that a triple within that set will still be available*
  - Set<Statement> **rd**(String **query**, URI **space**, int **timeout**)
    - like CoreAPI.rd, but supports SPARQL queries
  - Set<Statement> **rd**(String **query**, int **timeout**)
    - as above, without space URL
    - system can select a match from anywhere in Triple Space
  - Set<Set<Statement>> **rdmultiple**(String **query**, URI **space**, int **timeout**)
    - like rd, but returns multiple matches of the given templatematch may be set of triples, e.g. Concise Bounded Description
    - no completeness guarantee: the response to the client may not contain all matches within the space

Extended TS API - Subscriptions

- interface org.tripcom.api.ts.ExtendedAPI
  - URI **subscribe**(String **query**, Callback **callback**, URI **space**)
    - established a subscription via a callback (javax.security.auth.callback.Callback)
    - when a set of triples matching the template is written into the specified space, the matched set of triples is sent to the callback
    - returns a URI identifying the subscription
  - void **unsubscribe**(URI **subscription**)
    - cancels the subscription with the given URI if it exists

Further Extended TS API - in

- interface org.tripcom.api.ts.FurtherExtendedAPI
  - Set<Statement> **in**(String **query**, URI **space**, int **timeout**)
    - like rd, but removes the matched triples
  - Set<Set<Statement>> **inmultiple**(String **query**, URI **space**, int **timeout**)
    - like rdmultiple, but removes the matched triples

Further Ext. TS API - Transactions

- interface org.tripcom.api.ts.FurtherExtendedAPI
  - URI **createTransaction**(String **type**)
    - type: „local“ to a client (URI is only known to the client)
    - type: „shared“: transaction shareable with others.
      - Identifiers of shared transactions would be made available to other clients, who can then „get“ the shared transaction to participate in it
    - Transactions may be supported optimistically or pessimistically
      - pessimistically would potentially make less guarantees
  - boolean **beginTransaction**(URtransactionID)
    - begins the transaction
    - all subsequent interactions by all clients sharing this transaction are handled transactionally
  - boolean **commitTransaction**(URI **transactionID**)
    - commits all interactions made within this transaction in the space
  - boolean **rollbackTransaction**(URI **transactionID**)
    - rolls back all interactions made within this transaction in the space
  - boolean **getTransaction**(URI **transactionID**)
    - join in a shared transaction with other clients

Management API

- interface org.tripcom.api.management.ManagementAPI
  - public URI **create**(String **path**, URI **space**);
  - public void **destroy**(URI **space**);
  - public Set<URI> **children**(URI **space**);
  - public void **addMetadata**(Set<Statement> **metadata**, URI **space**);
  - public boolean mOut(Set<Statement> **policies**, URI **space**);

Example program

```
import org.openrdf.model.*;
import org.tripcom.api.ts.*;

// create instance of the CoreAPI wita a valid CERTIFICATE
CoreAPI tsapi = new CoreAPIImplementation(CERTIFICATE);

// create RDF triple
ValueFactory factory = new ValueFactoryImpl();

URI subject =
   factory.createURI("http://tripcom.sourceforge.net/tutorial");
URI predicate =
   factory.createURI("http://purl.org/dc/elements/1.1/title");
Literal object =
   factory.createLiteral(“TripCom Tutorial");
```

Example program (cont.)

```
Statement statement = new StatementImpl(
   subject, predicate, object);

// write the triple into a space
tsapi.out(statement, new
   URIImpl("tsc://www.tsexample.com/"));

// Create a simple query
String query = "SELECT * WHERE { ?x ?y ?z . }";

// read triples from the space
Set<Statement> result = tsapi.rd(query, new
   URIImpl("tsc://www.tsexample.com/"), 30000);

Sytem.out.println(result);
```

Kernel Installation

Required software

- Java JDK version 1.5
  - http://java.sun.com/javase/downloads/index_jdk5.jsp
- Apache Maven version 2.0
  - http://maven.apache.org/
- Subversion version 1.4
  - http://subversion.tigris.org/

Getting the source / Compile

- Getting the source
  - Subversion checkout

svn co \ https://tripcom.svn.sourceforge.net/svnroot/tripcom

- Compiling the source

cd tripcom/trunk

mvn clean package assembly:assembly

  - skipping the tests

(-Dmaven.test.skip=true)

Dependencies

- TripCom repository
  - passwort protected
- ~/.m2/settings.xml

```
<settings>
  <servers>
    <server>
      <id>TripComRepository</id>
      <username>tripcom</username>
      <password>TP!56tsc</password>
    </server>
  </servers>
</settings>
```

Using the repository

- pom.xml

```
<project>
  ...
  <repositories>
    <repository>
    <id>TripComRepository</id>
    <url>http://tripcom.sourceforge.net/repository</url>
    </repository>
    ...
  </repositories>
  ...
</project>
```

TripCom assembly

- tripcom-0.0.1-bin.zip

/libs Third party libraries

/bin TripCom jars

/conf configuration files

start.[sh|bat] start scripts

Applications

- jini

  http://www.jini.org/wiki/Category:Getting_Started

- blitz

  http://sourceforge.net/project/showfiles.php?group_id=126322&package_id=138181

  - installer_pj_2_0-rc4.jar

Starting the kernel

- starting blitz

sh blitz.sh

- removing the log/database file

rm -rf log/dbfiles_dancres/*

- starting the kernel

sh start.sh