Cultural and historical digital libraries dynamically mined from news archives

TrenDSTMap: The TrenDS Mapping Tool

Project Reference No.	**FP7-215874**
Deliverable No.	
Workpackage no:	**WP3: Ontology Modeling Mapping and Matching**
Nature:	**O (Other)**
Dissemination Level:	**PU**
Document version:	**2**
Date:	**10/09/2009**
Editor(s):	**Nadzeya Kiyavitskaya**
Document description:	Ontology Mapper/Editor Manual

History

Version	Date	Reason	Revised by
01	10/09/2009		
02	07/01/2010	Adding the evolution description	Siarhei Bykau

Authors List

Organisation	Name
UTR	Yannis Velegrakis
UTR	Nadzeya Kiyavitskaya
UTR	Siarhei Bykau
UTR	Chrisa Tsinaraki

Table of Contents

1. Introduction: Main Goal and Assumptions

TrenDSMap is intended to facilitate the editing of data repositories (including ontology repositories) and allow the cross-repository mapping creation. An online demo of TrenDSMap can be found at http://db.disi.unitn.it:8282/entitymanager

The terminology used in this document is mainly adopted from the fields of conceptual modeling and databases.

Glossary of adopted terms:

Association - same as *relationship*
Attribute - same as property
Data Repository - physical storage of ontologies
Entity - can be a concept or an instance
Individual - same as *instance*
Instance - the basic or "ground level" objects
Mapping - given two ontologies O1 and O2, mapping one ontology onto another means that for each entity in ontology O1, we try to find a corresponding entity, which has the same intended meaning, in ontology O2
Ontology - a formal representation of a set of concepts within a domain and the relationships between those concepts
Property - parameters that objects (and concepts) can have
Relationship - a way in which concepts and instances can be related to one another
Source - the ontology O1 in *Mapping*
Target - the ontology O2 in *Mapping*

2. Underlying Data Model

The main assumption is that the tool does not distinguish between concepts and instances, these two types of elements are called *entities* and displayed in the same way. It means that the results provided to the user can contain both concepts and instances. Hence, a mapping can be established between: concepts, instances, a mix of concepts and instances, both on the source and target sides.

Another assumption of the Mapper is that the tool is used mainly for the *creation and revision of mappings*, while browsing facilities are provided by the Browser in a separate window. Thus, the user should first study Papyrus ontologies in the Browser, find interesting candidates for creating a mapping, then proceed with selecting these entities in the Mapper interface and saving a new mapping.

The user should note that the tool embodies functionalities of both a mapper and an ontology editor. Therefore, in Section 2 we describe the elements that are shared between both instruments, while Section 3 provides an insight on the Mapping process and Section 4 explains in more detail the editing features.

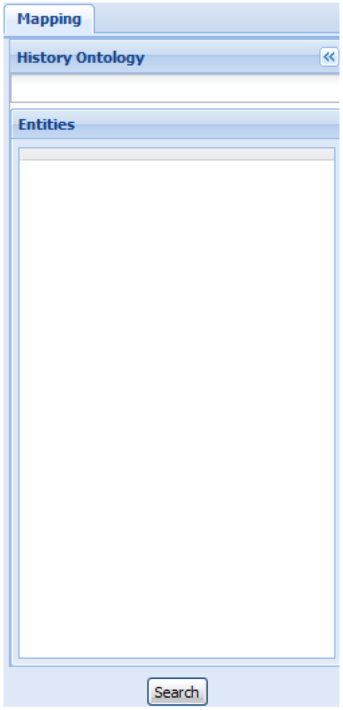
3. Description of Panels

This section provides general description of each panel in the user interface of the tool.

3.1. Source and Target search panels

These panels are intended for looking up entities of the source and target ontologies.

On the left-hand side you can browse the *History* ontology, while the *News* ontology is available on the right-hand side. The "Entities" boxes on both panels can show maximum 250 elements from the retrieved results. Therefore, the exact name of the entity must be first looked up in the Browser, and then entered in the Mapper's search fields.



(a) Source search panel is placed on the left hand side

News Ontology
Entities
Search

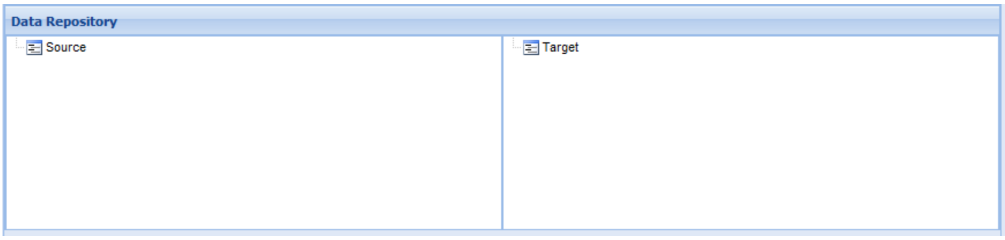
(b) Target search panel is placed on the right hand side

3.2. Data Repository window

The Data Repository window is used for representing and editing the content of entities retrieved as a result of a search operation. The user can drag one or more entities from the source and target ontologies to the left- and right-hand panels respectively, then explore the related attributes and relationships (also called *associations*) of these entities.

The Data Repository window is mainly intended for editing the ontologies. It allows the user to create, modify and remove entities, attributes or associations by using the appropriate options from the right mouse click menu.

In order to open an entity in the source or target windows of the Data Repository, the user should first look for this entity using the appropriate search panel, then drag the retrieved entity in the Data Repository window. The user may open one or more entities, each newly introduced entity will be added in the end of the list. In order to clear the window, the user should right click on the Source or Target label and select "Clear View".


Data Repository

Source

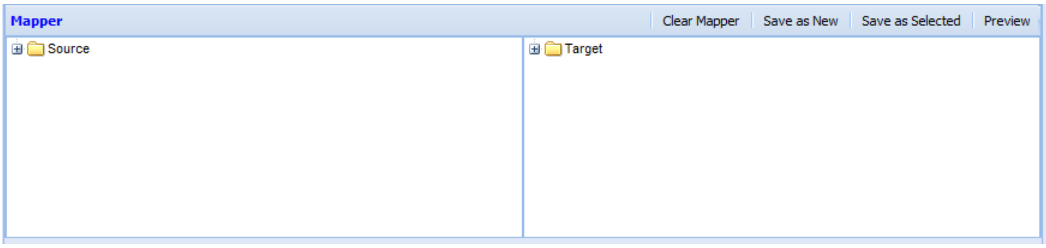
Target

3.3. Mapper window

After having a look at the candidate entities in the Data Repository window, the user may proceed with mapping generation by dragging one or more entity from source and target candidates to the Mapper window.

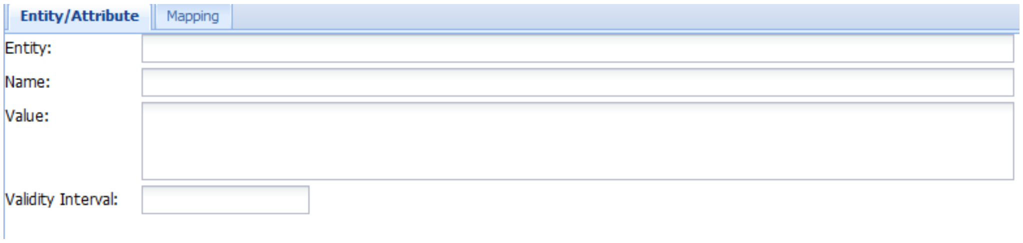
The window has 4 buttons:

- *Clear Mapper* - clears source and target windows of the Mapper view
- *Save as New* - saves the mapping with a new identifier
- *Save as Selected* - saves a modified version of the selected mapping, keeping its current identifier
- *Preview* - generates a preliminary mapping specification that can be viewed in the bottom most window (with orange background).



3.4. Entity/Attribute Tab

This is an information tab where the user can view all the attribute values of the entities selected in the Data Repository window.


Entity/Attribute | Mapping

Entity:

Name:

Value:

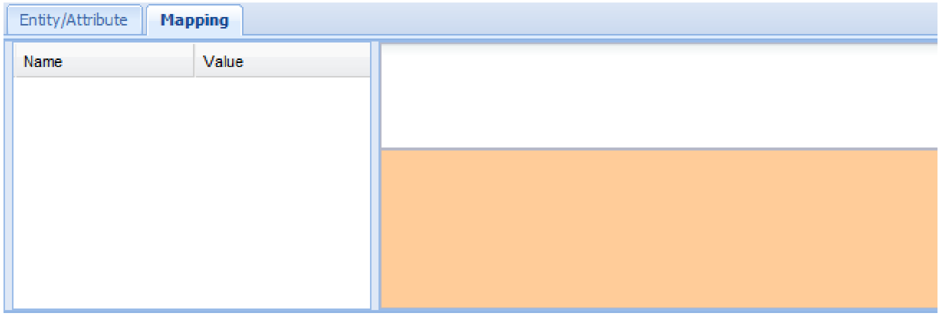
Validity Interval:

3.5. Mapping Tab

This is an information tab where the user can see the full list of mappings already created and, if necessary, revise an existing mapping.

On the right hand side of the mappings table, the user can see two horizontal windows:

- the upper window, with white background, allows to insert comments;
- the bottom window, with orange background, displays the formal mapping specification generated automatically by the tool during the save or the preview operation.



NB: We strongly advise users to document their mappings as much as possible, so that a mapping expert can manually check that the actual semantics of the new mappings corresponds to the intended meaning.

4. Guidelines to the Mapper

This section shortly provides general descriptions of the functional elements, explains the process of building a mapping, and uses a running example to illustrate this process.

The work with the tool envisions two main phases:

1 – Selecting candidate entities for both target and source ontologies;

2 – Creating, refining and saving a mapping;

3 – Editing existing mappings.

4.1. Selecting Candidates for Target and Source

Before building a mapping, a user is supposed to select candidates for both Target and Source ontologies:

- *Search in ontologies.* The search panels on left- and right-hand sides are intended for looking up entities of an ontology.
- *Previewing entities.* When you would like to preview a retrieved entity, drug-and-drop it to a corresponding source or target panel of the Data Repository window.

4.2. Creating a New Mapping

For each new mapping, the user must specify:

- source entities,
- target entities,
- and optionally, conditions.

After previewing the entities in the Data Repository panel, the user should drag-and-drop one or more entity in both source and target windows of the Mapper window.

4.3. Working with Conditions

The user can specify varied types of conditions for a mapping for on the source and target sides.

In order to:

- *not apply any conditions*, the user has to click "Clear attributes" from the right mouse click menu.
  Example: any Person from ontology 1 corresponds to any Human in ontology2, Person() → Human().
- *apply exact match condition on some attribute,* leave the value of the specific attribute filled and clear all other attributes from the tree.

  Example: Person from ontology 1 with Gender that equals to M must correspond to Male in ontology 2, Person(Gender="M") → Male().
- *use variables in condition,* the user shall delete the value of the specific attribute (double click allows editing) and put random variable name prefixed with dollar symbol "$". If you want to assign some condition on the value of this variable, then create condition under "Condition" label in the tree and assign your expression to the variable of your choice.

  Example: Person in ontology 1 of Gender equal to "F" and working in the theater corresponds to the Actress concept of ontology 2, Person(Gender="F", works_in=$x where $x="theater")→Actress()

- *express AND-condition,* the user shall simply select several entities on the corresponding mapping side.

  Example: Person in ontology 1 corresponds to both the Male and Female concepts of ontology 2, Person()→Male() AND Female()

- *express OR-condition* is not possible, however, such conditions can be partially simulated by creating two mappings.

  Example: Location in ontology 1 may contain Site or Place concepts of ontology 2, Location()→Site() ; Location()→Place().

NB: The most common case for most of the mappings is the one without conditions: just use the right-click menu to clear all attributes, thus applying no condition.

4.4. Modifying or Removing an Existing Mapping

In case you want to find and revise some mapping created earlier, you should:

- find a mapping of interest in the table of the Mapping Tab,
- double click on the ID of the mapping,

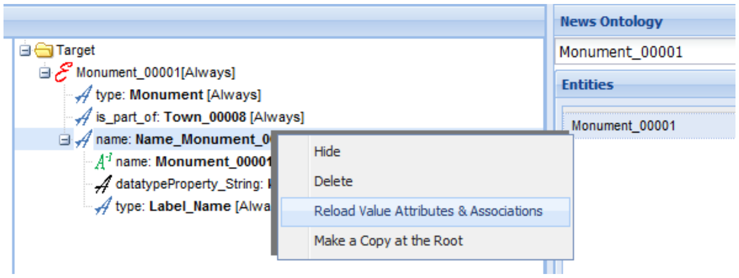
The mapping and its formal specification will be then opened in the Mapper window. You can insert your revisions and save the updated mapping.

Useful remarks when exploring two ontologies

History ontology: concepts of interest for the mappings are placed in *E1.CRM_entity* subtree

News ontology: concepts of interest for the mappings are placed in the following subtrees - (a) Entity, (b) ConceptEvaluation, (c) Concept->Theme->PapyrysTheme

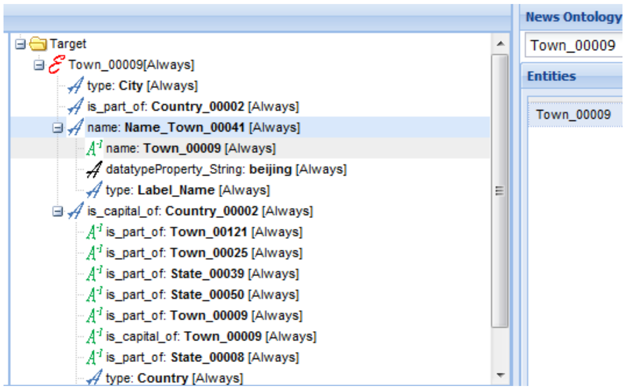
You may explore relationships of the chosen entity in Data Repository window, by clicking "Reload value attributes and associations" from the right-click menu. For example, when retrieving an entity *Monument_00001*, we can see that it has a name property that refers to another entity *Name_Monument_00182*. In order to know its actual value, we can click on "Reload value attributes and associations" and find out that the attribute *datatypeProperty_String* contains "kremlin" value.



Target
Monument_00001[Always]
type: **Monument** [Always]
is_part_of: **Town_00008** [Always]
name: **Name_Monument_0**
name: **Monument_00001**
datatypeProperty_String:
type: **Label_Name** [Alwa

News Ontology
Monument_00001
Entities
Monument_00001

Similarly, when looking at the entity *Town_00009*, one may use "Reload ..." option on *name* and *is_capital_of* properties to discover that this city is "Beijing", the capital of Country_0002, whose name results to be "China".



News Ontology

Town_00009

Entities

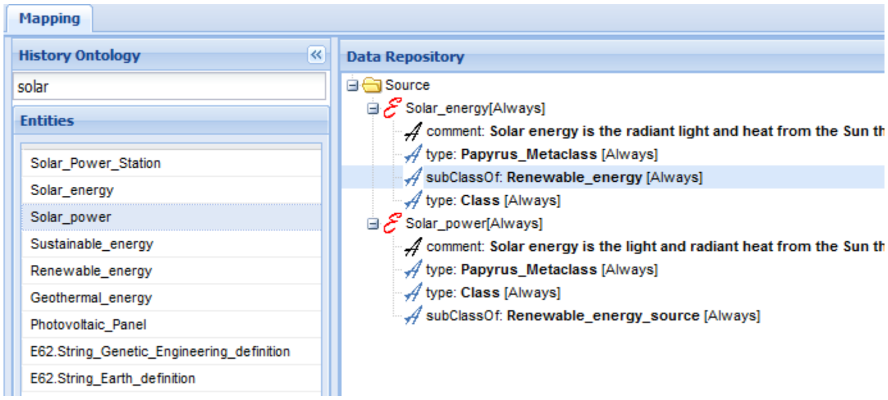
Town_00009

- Target
  - Town_00009[Always]
    - type: **City** [Always]
    - is_part_of: **Country_00002** [Always]
    - name: **Name_Town_00041** [Always]
      - name: **Town_00009** [Always]
      - datatypeProperty_String: **beijing** [Always]
      - type: **Label_Name** [Always]
    - is_capital_of: **Country_00002** [Always]
      - is_part_of: **Town_00121** [Always]
      - is_part_of: **Town_00025** [Always]
      - is_part_of: **State_00039** [Always]
      - is_part_of: **State_00050** [Always]
      - is_part_of: **Town_00009** [Always]
      - is_capital_of: **Town_00009** [Always]
      - is_part_of: **State_00008** [Always]
      - type: **Country** [Always]

4.5. Running Example

Let assume we decided to create a mapping between *Solar_energy* from the History ontology and *PapyrusTheme_SolarEnergy* from the news ontology.

The process starts with identifying those concepts in the source and target panels of the Data Repository. For instance, by entering "solar" in the History ontology search panel, the user may retrieve several candidates.



Mapping

History Ontology

solar

Entities

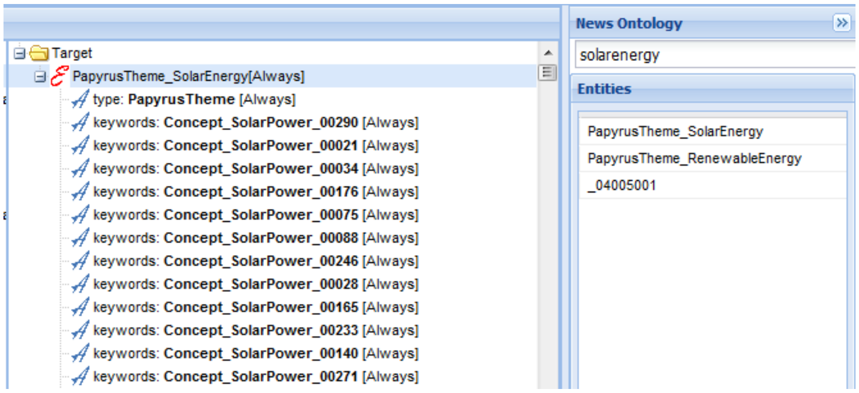
- Solar_Power_Station
- Solar_energy
- Solar_power
- Sustainable_energy
- Renewable_energy
- Geothermal_energy
- Photovoltaic_Panel
- E62.String_Genetic_Engineering_definition
- E62.String_Earth_definition

Data Repository

- Source
  - Solar_energy[Always]
    - comment: **Solar energy is the radiant light and heat from the Sun th**
    - type: **Papyrus_Metaclass** [Always]
    - subClassOf: **Renewable_energy** [Always]
    - type: **Class** [Always]
  - Solar_power[Always]
    - comment: **Solar energy is the light and radiant heat from the Sun th**
    - type: **Papyrus_Metaclass** [Always]
    - type: **Class** [Always]
    - subClassOf: **Renewable_energy_source** [Always]

The most interesting entities from the list of results seem to be *Solar_energy* and *Solar_power.* By dragging these entities in the Data Repository, we can see that *Solar_power* is a kind of energy source (looking at the *subClassOf* association), while *Solar_energy* is a type of the energy produced from that source. So, we decide to go for the *Solar_energy* and drag this entity into the Mapper window under the Source label.

After that, we proceed with selecting a target entity by looking for "solarenergy" in the News ontology search panel. We then drag PapyrusTheme_SolarEnergy into the Data Repository window, or directly under the Target label of the Mapper window, given that we are sure in our choice and don't need to explore the relationships of this entity.

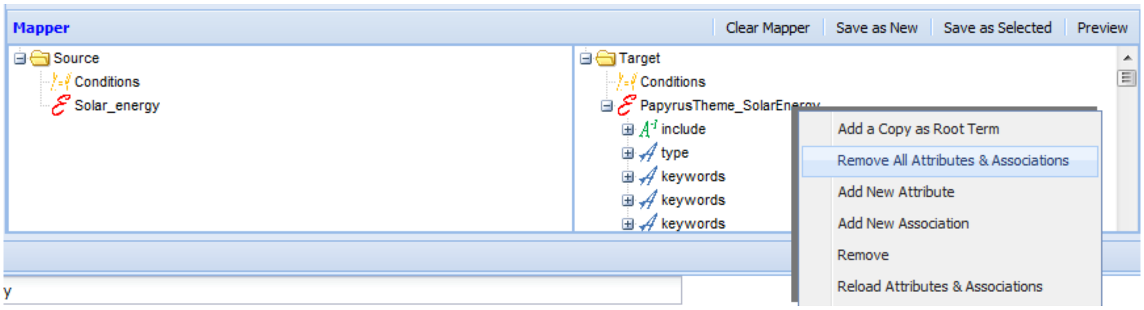


Target
PapyrusTheme_SolarEnergy[Always]
type: PapyrusTheme [Always]
keywords: Concept_SolarPower_00290 [Always]
keywords: Concept_SolarPower_00021 [Always]
keywords: Concept_SolarPower_00034 [Always]
keywords: Concept_SolarPower_00176 [Always]
keywords: Concept_SolarPower_00075 [Always]
keywords: Concept_SolarPower_00088 [Always]
keywords: Concept_SolarPower_00246 [Always]
keywords: Concept_SolarPower_00028 [Always]
keywords: Concept_SolarPower_00165 [Always]
keywords: Concept_SolarPower_00233 [Always]
keywords: Concept_SolarPower_00140 [Always]
keywords: Concept_SolarPower_00271 [Always]

solarenergy

PapyrusTheme_SolarEnergy
PapyrusTheme_RenewableEnergy
_04005001

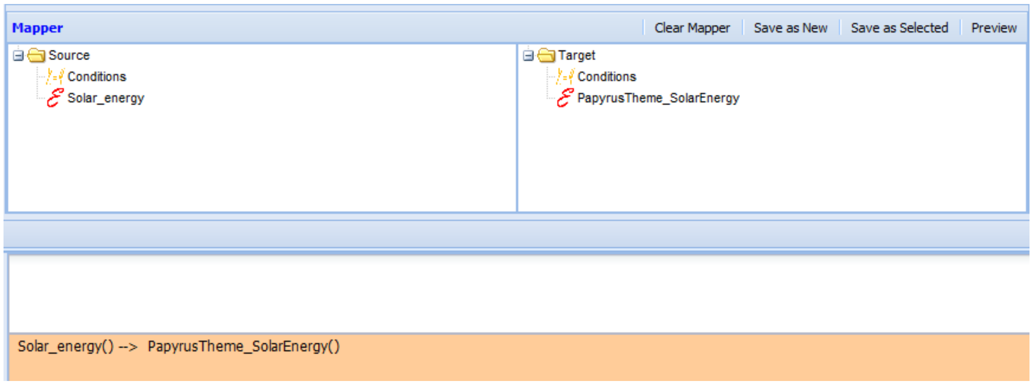
We don’t want to express any conditions on this mapping, therefore we decide to remove all attributes and associations both on the left and right hand sides.



Source
Conditions
Solar_energy

Target
Conditions
PapyrusTheme_SolarEnergy
include
type
keywords
keywords
keywords

We would to preview the syntax of this mapping before saving it, so we press Preview button and in see the generated piece of formal specification (switch to the Mapping tab on the left bottom side).

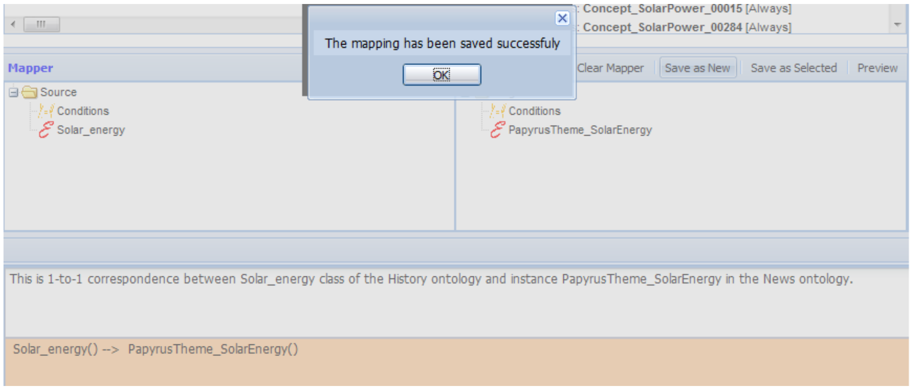


Source
Conditions
Solar_energy

Target
Conditions
PapyrusTheme_SolarEnergy

```
Solar_energy() --> PapyrusTheme_SolarEnergy()
```

Finally, we add a comment in the text box and press ”Save as New” button to register this mapping in the repository.



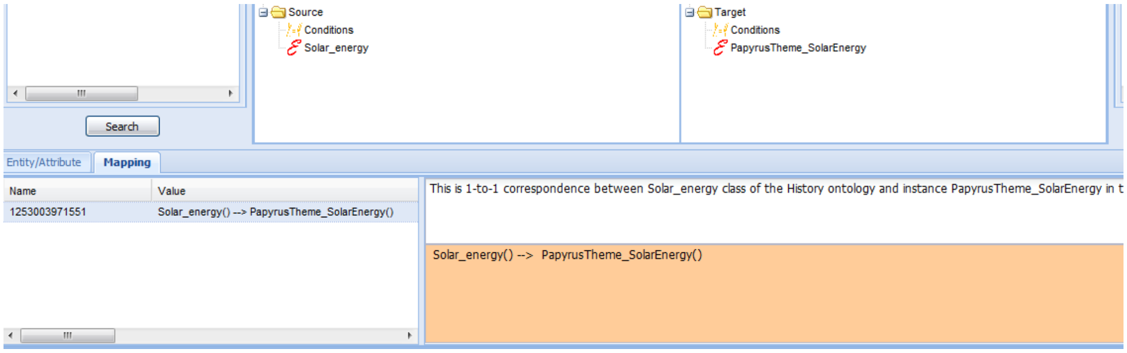
The mapping has been saved successfuly

This is 1-to-1 correspondence between Solar_energy class of the History ontology and instance PapyrusTheme_SolarEnergy in the News ontology.

Solar_energy() --> PapyrusTheme_SolarEnergy()

The confirmation message appears saying that our mapping has been created successfully. After that, we can clear the Mapper window using "Clear Mapper" button.

To go back to view or edit this mapping, we may find it in the Mapping Tab table, double-click on it, and the Mapper window will automatically display the content of the mapping.

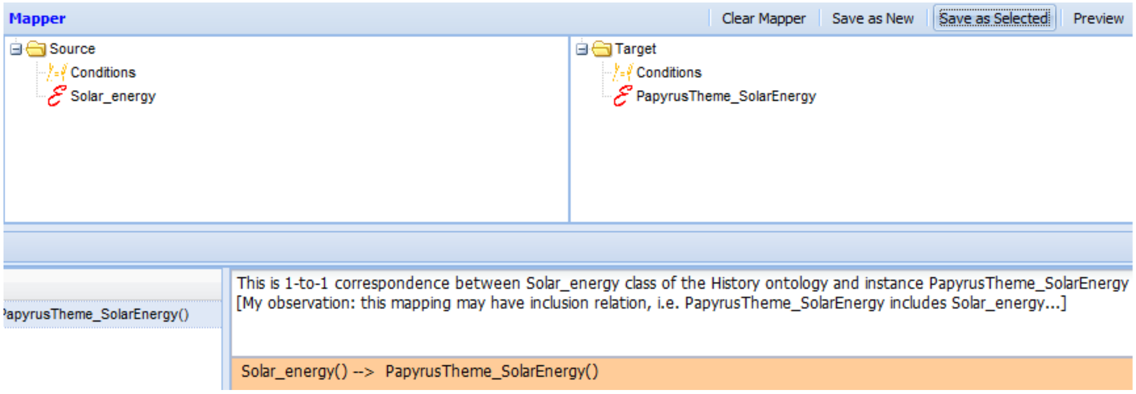


Name	Value
1253003971551	Solar_energy() --> PapyrusTheme_SolarEnergy()

This is 1-to-1 correspondence between Solar_energy class of the History ontology and instance PapyrusTheme_SolarEnergy in t

Solar_energy() --> PapyrusTheme_SolarEnergy()

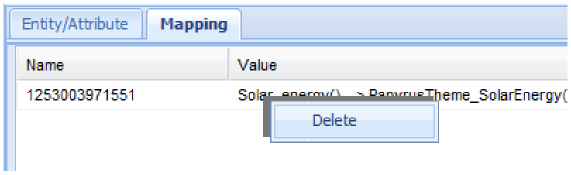
We add another observation in the comments window and press "Save as Selected" to update the mapping in the repository.



This is 1-to-1 correspondence between Solar_energy class of the History ontology and instance PapyrusTheme_SolarEnergy
[My observation: this mapping may have inclusion relation, i.e. PapyrusTheme_SolarEnergy includes Solar_energy...]

Solar_energy() --> PapyrusTheme_SolarEnergy()

We can remove the mapping at any time from the table (and the physical repository), by using the right-button mouse click option "Delete".



Name	Value
1253003971551	Solar_energy() --> PapyrusTheme_SolarEnergy(

5. Guidelines to the Editor

This section explains in detail the ontology editing functionalities provided by the tool.

5.1. Insert a new element

In the Data Repository window, right click on the "Source"/”Target” root of the tree opens a menu that contains the following options:

- "Clear Entities": removes all its children from the view.
- "New Entity": creates a new entity with a default name "default_Entity" root element of the tree.

5.2. Edit an element

Right clicking on an entity or a property in the tree opens a menu that contains several options:

- *Hide* eliminates the entity node and the subtree under it (but does not delete it from the repository).
- *Delete* deletes the entity from the view BUT ALSO from the RDF repository
- *Reload all attributes & associations* updates a list of all the attributes and associations of the specific entity from the repository.
- *Introduce new attribute* adds a new attribute under the entity in focus with a default name "default" and a default value "default". NOTE, that it is impossible to have two identical attributes for any entity (equal values in the equal attributes).

Right clicking on an attribute in the tree opens a menu that contains the options:

- *Introduce new association* adds a new association.

The tab Entities/Attributes provides a convenient way to edit the values of the entity attributes in the focus of the Data Repository window. Once the user finishes editing, a button "Update" should be pressed in order to refresh the view in the Data Repository window and the repository itself.

In addition, right clicking on an entity has also options:

1. *Reload all attributes and association* - updates the list of attributes and associations of the data repository
2. *Hide contained attributes and associations* - clears the list of attributes and associations from the view.

6. Temporal Support

6.1. Intervals

The data model used by TrenDS includes support for temporal information. Time is represented in terms of fuzzy interval. An interval is a continuous period in time represented by two time points: its beginning and its end. Intervals are represented by including the beginning and the end time in brackets, and separating them by a comma. For instance, the time period from the 11/11/1974 to 01/06/1978 is represented as [11/11/1974, 01/06/1978]. A fuzzy period is a period in which its beginning and its end are not exact points in time but they are themselves a period. For instance, assume that that we know that something started some time in fall 1993 and ended in spring 2000. This can be represented as:

```
[ [01/09/1993, 30/11/1993], [01/03/2000, 31/05/2000] ]
```

Actually, the accepted syntax for the dates is not following the standard dd/mm/yyyy format, but the yyyymmdd. This means that the correct syntax for the above fuzzy interval is

```
[ [19930901, 19931130], [20000301, 20000531] ]
```

An important note that needs to be made regarding this format is to never prefix the years with zeros but only the months and the days. For instance, the 1st of March of the year 643 should be represented as 6430301.

The day is the minimum granularity supported by TrenDS. Additional granularities that are supported are the decades, millenniums and centuries. The syntax is the one of a number followed by the specification. For instance, the 16th century is represented as: 16century.

Apart from the different cardinalities, the system supports also some predefined periods, like for example, the medieval time. In these cases just using the name of the period is enough. In the specific case, it is the word medieval.

Finally it is possible to avoid using intervals when there is no need. For instance, if we know that the beginning of a period is exactly the 1st of Sep of 1993, we can say:

```
[ 19930901, [20000301, 20000531] ]
```

where as you can notice the first part of the interval is not an interval but a specific date. Equally, we an even use an interval that has only one element. For instance we can say [medieval] or [21century].

Temporal specification can be associated either to entities, i.e., classes or individuals, or to attributes and associations. This is done by the special field on the interface called Interval and is located at the editing tab at the bottom of the screen.

If the field is left empty, then the default value [Always] is assumed which means that the lifespan has no limits.

6.2. Use of intervals in mappings

A temporal support is provided in mappings that allow users to specify mappings having constraints on temporal information. The temporal specification on queries or mappings has exactly the same syntax as the one provided in 6.1, with the addition that instead of constraint values, variable can be used. For instance, one can say [$x, 21century] which means some period that started at some point (specified by the variable $x) and finished in the 21st century.

An interval specification in a mapping can be made on any entity, right after the name. For instance, the following mapping, maps to Machines the entity computer of the 20th century.

```
Computer@[20century]() ---> Machines()
```

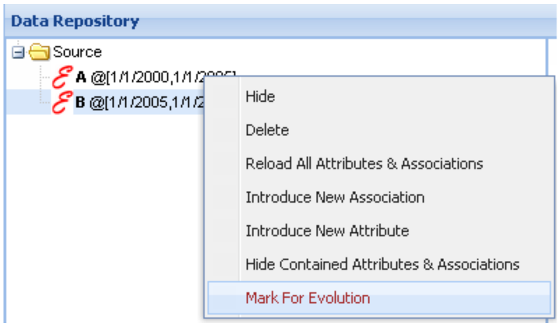
Similarly, time specification can also be done on the attributes/associations. The syntax is similar and is done right after the name of the attribute. For instance, the following mapping maps to the person called Obama who is living in the White House at some period in 2009 to the president of the United States.

```
$x(name: Obama, livesIn@[20090301,20091230]:WhiteHouse) --> President(country: US)
```

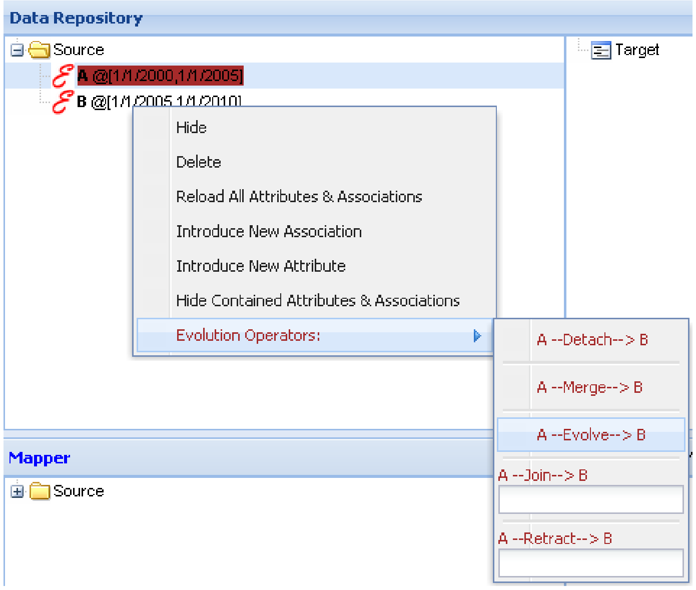
7. Evolution Support

7.1. Create an Evolution Operator

The management of evolution operators are supported by the Data Repository Windows (see Section 3.2). In order to create a new evolutional operator the user must choose two entities. The one, from which an evolution operator goes, and another one to which it follows. The first entity is picked by the item in context menu which appears in red for all entities in a data repository list.



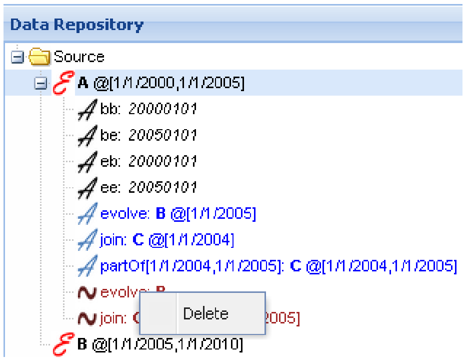
The next step is to choose the second entity and type of operator to be applied (note that the previous entity can be unmarked from being in the evolution operator by choosing "UnMark For Evolution"). This is done again through the context menu which now has the option of operators along with their parameters (for join and retract it is a time point).



When the operator is chosen the system will validate the intervals of entities and notify the user about the results (either successful or not). The evolutionary properties will appear for both entities marked in red accompanied by the temporal intervals.

7.2. Remove an Evolution Operator

The removal of evolution operators is supported by a context menu of evolution attributes which appear together with the normal ones. The user may choose the evolutional operator of interest and its contextual menu will contain "Delete" option.



Data Repository

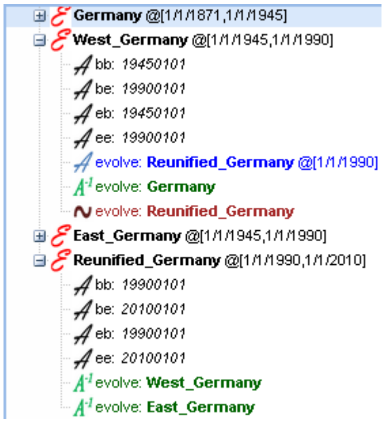
Source
- A @[1/1/2000,1/1/2005]
  - bb: 20000101
  - be: 20050101
  - eb: 20000101
  - ee: 20050101
  - evolve: B @[1/1/2005]
  - join: C @[1/1/2004]
  - partOf[1/1/2004,1/1/2005]: C @[1/1/2004,1/1/2005]
  - evolv B
  - join: C 2005]
- B @[1/1/2005,1/1/2010]

7.3. Running Example

To illustrate the possible use cases of evolution operators' management let assume that we have four entities which describe the history of Germany. They are *Germany[1871-1945], East Germany[1945-1990], West Germany[1945-1990]* and *Reunified Germany[1990-Now]*. We want to express the following evolutionary facts. *Germany* evolved into *East* and *West Germany* which later on in 1990 were again joined (evolved) to create *Reunified Germany*. The set of binary evolutionary operators which represent these facts is the following:

1. *Germany* evolve *East Germany*
2. *Germany* evolve *West Germany*
3. *East Germany* evolve *Reunified Germany*
4. *West Germany* evolve *Reunified Germany*

These four operators are created using steps described in Section 7.1. As a result we can see the evolutionary attribute of West Germany (in red) like it is shown in the figure:



- Germany @[1/1/1871,1/1/1945]
- West_Germany @[1/1/1945,1/1/1990]
  - bb: 19450101
  - be: 19900101
  - eb: 19450101
  - ee: 19900101
  - evolve: Reunified_Germany @[1/1/1990]
  - evolve: Germany
  - evolve: Reunified_Germany
- East_Germany @[1/1/1945,1/1/1990]
- Reunified_Germany @[1/1/1990,1/1/2010]
  - bb: 19900101
  - be: 20100101
  - eb: 19900101
  - ee: 20100101
  - evolve: West_Germany
  - evolve: East_Germany

All the attributes (except evolutionary ones) depicted in the figure are the internal representation of temporal and evolutionary data and for the end-user mode should be hidden.