Travelogue/Space Implementation Documentation

Koen Van der Auwera and Bob Van Landuyt

The online prototype Travelogue is an initiative of the European network SPACE (Supporting Performing Arts Circulation in Europe) that aims to link and exchange data on international touring in the performing arts using Linked Data technology. This text describes the reference implementation by VTi (Flemish Institute for the Performing Arts), which uses a D2R Server to publish the database on the Semantic Web. The document also explains how to publish your own database with D2R in such a way that the SPACE network is able to interpret your data in order to process and combine it with other data from multiple sources.

This version dates from October 2011. A more recent version can be consulted on www.arts-mobility.info.

space

supporting performing arts circulation in europe

Requirements

D2R Server

The following components are needed to install the D2R Server:

→ **Java 1.4** or newer. Check this via the command java -version if you are not sure.

→ A **supported database**. We use PostgreSQL, but D2R Server also supports MySQL, Oracle and Microsoft SQL Server. Other databases have not been tested, but any SQL-92 compatible database should work with minor configuration changes. An ODBC data source such as Microsoft Access also works, although with serious limitations. For example, the mapping generator does not work with ODBC.

→ Optionally, a J2EE servlet container as a deployment target. In this document, we run D2R Server as a stand-alone web server, which is the easiest to set up.

→ A **server**. We are running the prototype on a Linux server, more specifically Debian Lenny (5.0). Any server that runs Java and the database of your choice will work. This **D2R** server provides a SPARQL endpoint, which we will use to read and parse the data that is published.

The D2R server files are available here: http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/

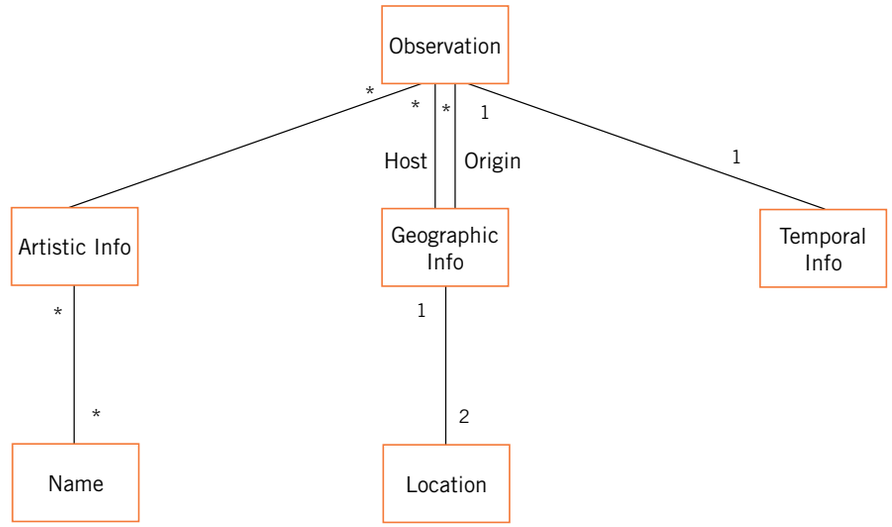
The Database

As described above, a supported relational database is required. If your data is not already contained in such a database, a database can be designed according to the data model described below. Your data can then be imported into the relational database and thus made available on the Semantic Web.

It is also possible to adjust the mapping with custom queries, or database views can be created to make the data available to the D2R server.

The Data Model

The project expects a specific data model and vocabulary in order to read and process your data. Below is the UML diagram of what needs to be made available by the D2R server. Additional vocabularies and properties may be added. They will not interfere with the data used by the SPACE project.



Observation

An observation (also called a “datapoint”) is the key entity of the data model: it assembles all the information on one specific stage production: who, what, where and when? Since not all datasets register this information with the same level of detail, different Observation objects can have different sets of properties. That is, not all properties need to be present in all Observation objects. The relationships between the observations host and origin geographic info, and its Artistic Info, are more important.

Property	Data type or value	Description
rdfs:label	String	A humanly readable and descriptive name for the observation.
rdf:type	<vocab:observations>	The RDF type of the observation.
vocab:observations_screen_display	String	The name of the observation as displayed to users on the screen, which is identical to the rdfs:label property of this object.
vocab:observations_nr_of_performances	xsd:int	The number of performances of this observation during the time of its Temporal info object. This is an integer greater than or equal to 0. Not all Observation objects have this property, and the value 0 means that the number of performances is unknown.
vocab:observations_nr_of_days	xsd:int	The number of days this stage, production was performed, which is an integer greater than or equal to 1. Not all Observation objects have this property.
vocab:observations_count_of_audiences	xsd:int	The number of people in the audience of this observation. This is an integer greater than or equal to 0. Not all Observation objects have this property, and the value 0 means that the number of persons in the audience is unknown.
vocab:observations_count_of_youth	xsd:int	The number of youngsters in the audience of this observation. This is an integer greater than or equal to 0. Not all Observation objects have this property, and the value 0 means that the number of youngsters in the audience is unknown.
vocab:observations_count_of_audiences_by_age_0_to_5	xsd:int	The number of people between the ages of 0 and 5 in the audience of this observation. Not all Observation objects have this property.
vocab:observations_count_of_audiences_by_age_6_to_15	xsd:int	The number of people between the ages of 6 and 15 in the audience of this observation. Not all Observation objects have this property.
vocab:observations_count_of_audiences_by_age_16_to_25	xsd:int	The number of people between the ages of 16 and 25 in the audience of this observation. Not all Observation objects have this property.
vocab:observations_occupation	xsd:int	The occupation of this stage production. Not all Observation objects have this property.
vocab:observations_school_performance	xsd:boolean	A Boolean flag defining whether or not this observation concerns a school performance. Not all Observation objects have this property.
vocab:observations_nr_of_people_on_tour	xsd:int	The number of people on tour to perform this stage production. Not all Observation objects have this property.
vocab:observations_financial_variant	String	The financial variant of this stage production. Not all Observation objects have this property.
vocab:observations_notes	String	Special notes about this observation. Not all Observation objects have this property.
vocab:observations_specifics	String	Specific notes about this observation, e.g. that it was not open to the public. Not all Observation objects have this property.
Relation	**Refers to**	
vocab:observations_artistic_infos	vocab:artistic_infos	The URI refering to the artistic info of this particular observation
vocab:observations_host_geographic_info	vocab:geographic_infos	Refers the geographic info object that exported this particular observation
vocab:observations_origin_geographic_infos	vocab:geographic_infos	Refers to the geographic info object this observation was exported to.
vocab:observations_temporal_infos	vocab:temporal_infos	Refers to the date and other temporal info for the observation.

Artistic Info

An Artistic info object represents all the artistic info concerning a certain observation.
This object has a relationship to all the people (artists, authors, producers) responsible for the observation.

Property	Data type or value	Description
rdfs:label	String	A humanly readable and descriptive name for the artistic info.
rdf:type	<vocab:artistic_infos>	The RDF type of the artistic info.
vocab:artistic_infos_screen_display	String	The name of the artistic info as displayed to users on the screen, which is identical to the rdfs:label property of this object.
vocab:artistic_infos_id	xsd:int	The identifier of the artistic info, which is a unique number among all Artistic info objects.
vocab:artistic_infos_production_type	String	The type of production for this Artistic info object, e.g. “performance”, “play”, “theatre”, “opera”, and so on. Not all Artistic info objects have this property.
vocab:artistic_infos_genre	String	The genre of this Artistic info object, e.g. “dance”, “musical”, “theatre”, “opera”, and so on. Not all Artistic info objects have this property.
vocab:artistic_infos_show_title	String	The title of the performance. If this property is empty, the observation will be ignored when counting.
vocab:artistic_infos_names	vocab:names	A name of one of the persons responsible for the performance, such as the artist or the producer. Not all Artistic info objects have this property, and an Artistic info object can have several of these properties, e.g. one for each artist and one for the producer.
vocab:artistic_infos_website	String	The website of this Artistic info object, e.g. the website of the artist. Not all Artistic info objects have this property.
vocab:artistic_infos_institution_type	String	The type of institution of this Artistic info object. Not all Artistic info objects have this property. Examples include “state theatre” or “independent theatre”.
Relation	**Refers to**	
vocab:artistic_infos_names	vocab:names	This property can occur multiple times. It points to the people who were involved with the production.

Name

A Name object represents the name and role of a person who is responsible for an Artistic info object, such as an artist or producer.

Property	Data type or value	Description
rdfs:label	String	A humanly readable description of the name.
rdf:type	<vocab:names>	The RDF type of the name.
vocab:names_name	String	A humanly readable description of the name, which is identical to the rdfs:label property of this object.
vocab:names_role	String	The role that this name plays, e.g. “producer”, “director”, “choreographer”, “composer”, “author”, “artist”, “conductor”, and so on. Not all Name objects have this property.

Geographic Info

A Geographic info object represents a location. The granularity of this location can vary greatly from a specific venue to an entire country. Because of this granularity, a Geographic info object links to one or more Location objects of different types. Ideally, one will be of the type “country” and one of the type “city”, but these relationships can be empty, as can the other properties of the Geographic Info object.

Geographic info has the following properties:

Property	Data type or value	Description
rdfs:label	String	A humanly readable name for the Geographic info object.
rdf:type	<vocab:geographic_infos>	The RDF type of the Geographic info object.
vocab:geographic_infos_screen_display	String	The name of the Geographic info object as displayed to users on the screen, which is identical to the rdfs:label property of this object.
vocab:geographic_infos_venue	String	The name of the venue. Not all Geographic info objects have this property.

vocab:geographic_infos_website	String	The website of the Geographic info object, e.g. the website of the festival. Not all Geographic info objects have this property.
vocab:geographic_infos_festival	String	The name of the festival, if this object refers to a festival location. Not all Geographic info objects have this property.
vocab:geographic_infos_continent	String	The name of the continent of this location. Not all Geographic info objects have this property.
Relation	**Refers to**	
vocab:geographic_infos_locations	vocab:locations	A location where this Geographic info object is located. This property can occur multiple times, for example to link the Geographic info object to locations of different granularity, such as one location with the city and one location with the country.

Location

A location represents a specific type of place, for example a city or a country.
Location has the following properties:

Property	Data type or value	Description
rdfs:label	String	A humanly readable name for the location.
rdf:type	<vocab:locations>	The RDF type of the location.
vocab:locations_name	String	The name of the location, which is identical to the rdfs:label property of this object.
vocab:locations_type	String	The type of location. Possible values for this property are “City” or “Country”.

Temporal info

A Temporal info object represents a time that can be granular to different levels: a date, a month, a year, a season or a generic time span from one particular date to another one. For example, a date is represented as a time span with identical start and end date. Because of these varying levels of granularity, not all properties need to be present in all Temporal info objects. Please note that the year property is important for grouping the data.

The following properties are available:

Property	Data type or value	Description
rdfs:label	String	A humanly readable name for this Temporal info object. For a season this is in the form “yyyy/zzzz”, for a year “yyyy”, for a month “mm/yyyy”, for a date “dd/mm/yyyy”, and for a generic time span. This is in the form “dd – ee/mm/yyyy”, “dd/mm – ee/nn/yyyy” or “dd/mm/yyyy – ee/nn/zzzz”.
rdf:type	<vocab:temporal_infos>	The RDF type of the Temporal info object.
vocab:temporal_infos_screen_display	String	The name of the Temporal info object as displayed to users on the screen, which is identical to the rdfs:label property of this object.
vocab:temporal_infos_start_at	String	The time of day at which the Temporal info object starts, in the format “hh:mm”.
vocab:temporal_infos_day	xsd:int	The day the time span starts. This must be a number from 1 to 31 that is a valid day in the month vocab:temporal_infos_month of the year vocab:temporal_infos_year. This property is present if and only if the property vocab:temporal_infos_end_day is present. Moreover, if this property is present, the property vocab:temporal_infos_month also must be present. Not all Temporal info objects have this property.
vocab:temporal_infos_end_day	xsd:int	The day the time span ends. This must be a number from 1 to 31 that is a valid day in the month vocab:temporal_infos_end_ month of the year vocab:temporal_infos_end_year. This property is present if and only if the property vocab:temporal_infos_day is present. Not all Temporal info objects have this property.
vocab:temporal_infos_month	xsd:int	The month the time span begins. This must be a number from 1 to 12. This property is present if and only if the property vocab:temporal_infos_end_m onth is present. Moreover, if this property is present, the property vocab:temporal_infos_year also must be present. Not all Temporal objects have this property.

Mapping the database

The D2R server uses a .n3 mapping file to query data from the database and make it available on the Semantic Web. A default mapping file will be made available with all de classMaps (the different models described above) and their properties. The mapping file then needs to be adapted to suit your database and server configuration.

Configuration

First, a number of prefixes need to be configured. These prefixes are used throughout the mapping and contain information on the mapping and the vocabularies used. If you wish to add additional information in other vocabularies, these can be defined here. In the mapping for SPACE, we only need to complete two prefixes:

```
@prefix map: <file:name-of-your-mapping.n3#> .
@prefix vocab: <http://your-host/vocab/resource/> .
```

The map prefix refers to our mapping file. The second “vocab” prefix tells the D2R server where the descriptions of the different (custom) vocabularies can be found. This allows us to use “vocab:observations” to refer to the class description of, for example, observations. By default, SPACE expects its resource definitions to be at the URL “/vocab/resource”. If a different location is used, this needs to be configured. Then we can add the configuration for our database:

```
map:database a d2rq:Database;
   d2rq:jdbcDriver “org.postgresql.Driver”;
   d2rq:jdbcDSN “jdbc:postgresql:database-name”;
   d2rq:username “your-username”;
   d2rq:password “your-pass”;
```

The first setting “jdbcDriver” specifies the driver we are using, in this case PostgreSQL. The default is MySQL, so if you are using MySQL, there is no need to include this property. The next setting specifies the database to be connected to. Together with the username and password, this allows the connection to the database to be made. The server itself can be configured as follows:

```
<> a d2r:Server;
   rdfs:label “Label for your server”;
   d2r:baseURI <http://your-host/>;
   d2r:port 80;
   d2r:vocabularyIncludeInstances true;
```

The label property sets the title of the web interface for the D2R server. The baseURI is used as base for all the “uriPatterns” used in the mapping. The port specifies the port on which the server is made available. The last property indicates whether or not the type is included in the several classmaps. Although not strictly necessary, it is recommended to leave this set to “true”, which is why the description of each classmap is included.

Mapping the database

When mapping a class, we first must define it using the following declaration:

```
map:locations a d2rq:ClassMap;
   d2rq:dataStorage map:database;
   d2rq:uriPattern “locations/@@production.locations.permalink@@”;
   d2rq:class vocab:locations;
   d2rq:classDefinitionLabel “locations”;
```

After the “map:” prefix, the name of the class is defined. The uriPattern refers to the URI where the objects can be found, which is also based on the “baseURI” setting that was specified earlier. Make sure that the value used in the URI is unique for every record in the database, otherwise only the first record will be available. The class property sets the vocabulary used. The classDefinitionLabel is a humanly readable name for the class. Each class needs to be defined in this way before properties can be assigned.

A label can also be defined for each class. This label ensures that humanly readable links are available in the D2R web interface. When there is no label specified for a class, the “uriPattern” is used. A label can be specified as follows:

```
map:locations__label a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:locations;
   d2rq:property rdfs:label;
   d2rq:pattern “@@production.locations.name@@”;
```

Properties for the classes can be defined as follows:

```
map:locations_name a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:locations;
   d2rq:property vocab:locations_name;
   d2rq:propertyDefinitionLabel “locations name”;
   d2rq:column “production.locations.name”;
```

If your database differs from the model described and the value cannot be retrieved from a column, it is also possible to write JOIN properties such as this:

```
map:temporal_infos_start_at a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:temporal_infos;
   d2rq:property vocab:temporal_infos_start_at;
   d2rq:propertyDefinitionLabel “temporal_infos start_at”;
   d2rq:join “production.shows.date_id = production.date_isaars.id”;
   d2rq:column “production.shows.time”;
```

If necessary, more complex queries can be made to the database:

```
map:observations_nr_of_people_on_tour a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:observations;
   d2rq:property vocab:observations_nr_of_people_on_tour;
   d2rq:propertyDefinitionLabel "observations nr_of_people_on_tour";
   d2rq:join "production.shows.organisation_id =
production.organisations.id";
   d2rq:join "production.alumni.organisation_id =
production.organisations.id";
   d2rq:sqlExpression "COUNT(DISTINCT(production.alumni.person_id))";
   d2rq:datatype xsd:int;
```

It is also possible to define SQL "where" conditions using "d2qr:condition". If your database structure differs too much, it is also possible to create database views and obtain the necessary data from these views.

Relationships between the different classmaps can be defined as follows:

```
map:observations_artistic_info a d2rq:PropertyBridge;
   d2rq:belongsToClassMap map:observations;
   d2rq:property vocab:observations_artistic_info;
   d2rq:refersToClassMap map:artistic_infos;
   d2rq:join "production.artistic_infos.production_id =
production.shows.production_id";
```


Useful links

D2R Server: http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/

D2RQ mapping language: http://www4.wiwiss.fu-berlin.de/bizer/d2rq/spec/

SPARQL query language: http://www.w3.org/TR/rdf-sparql-query/

About SPACE

Ten national cultural institutions with an international policy and practice have created a new platform dedicated to Supporting Performing Arts Circulation in Europe: SPACE.

The members of SPACE occupy a position between politics and the artistic field in their countries, work as information centres, promote the (performing) arts at national and international level, and are experienced in supporting and running European cultural projects.

They share the belief that one of the cornerstones of European Cultural Policy is facilitating the circulation of (performing) arts across Europe, and realise there are still many imbalances in this transnational arts sphere among countries, regions, artists, disciplines and cultural operators.

The SPACE project's priorities include the mobility of arts productions and the combination of cultural mobility with cultural diversity, European citizenship, and investing in upcoming generations. Still a young initiative, SPACE intends to enlarge the network while implementing the different activities of the multi-layered project.

Members
ONDA (*Office National de Diffusion Artistique*), Paris
VTi (*Vlaams Theater Instituut*), Brussels
TIN (*Theater Instituut Nederland*), Amsterdam
NTIL (New Theatre Institute of Latvia), Riga
British Council, London
MIBAC, (*Ministero Beni e Attività Culturali*), Rome (which took over after ETI (*Ente Teatrale Italiano*) was shut down in June of 2010)
Pro Helvetia, Zürich
The Red House, Sofia
Institut umění - Divadelní ústav (Arts and Theatre Institute), Prague
Zentrum BRD des Internationalen Theaterinstituts, Berlin

Partners
ENICPA (*European Network of Information Centres for the Performing Arts*)
IETM (*International Network for Contemporary Performing Arts*)
La Belle Ouvrage
TEAM Network

About the authors and the designer

Koen Van der Auwera gained experience as a developer at a number of large and small companies and organisations. In 2006, he co-founded 10to1, and since then has been developing customised mobile and web applications. As CTO, he is responsible for the team of developers and the project planning.

Bob Van Landuyt (10to1) finished his studies at the Hogeschool West-Vlaanderen in January 2011. He started immediately as a web developer for 10to1. He quickly earned his spurs in this area and was able to broaden his knowledge to include iOS and Android development.

Gunther Fobe studied graphic design at the Saint-Lucas Institute in Ghent. For the first ten years of his graphics career, he worked for the multilingual communication office Poplar and the Carbon 7 Records label, both based in Brussels. He is now working as an independent graphic designer in Ghent. He is currently 'in-house' designer for – amongst others – VTi, Courtisane Festival and Arthouse Cinema Sphinx.

Colophon

This document is part of the publication Travelogue. Mapping Performing Arts Mobility in Europe.
http://www.arts-mobility.info

ISBN 9789074351416
D/2011/4610/2

http://www.arts-mobility.info
http://www.spaceproject.eu

Concept: Joris Janssens, Bart Magnus, Dries Moreels, Ann Olaerts @ VTi
Technical development: 10to1
Editorial staff: Joris Janssens, Bart Magnus, Koen Van der Auwera, Bob Van Landuyt
Translation and copy editing: Dan Frett
Production: Marijke De Moor
Layout publication: Gunther Fobe
Data visualisation: Pierre Huyghebaert & Pierre Marchand (Speculoos & OSP - Constant vzw)

Brussels, October 2011

space
supporting performing arts circulation in europe

Education and Culture DG
Culture Programme