STI INNSBRUCK

TOURISMUSVERBAND CONTENT ONTOLOGY MAPPINGS IMPLEMENTATION

Zaenal Akbar, Christoph Fuchs, Serge Tymaniuk, Ioan Toma
STI Innsbruck, University of Innsbruck,
Technikerstraße 21a, 6020 Innsbruck, Austria
firstname.lastname@sti2.at

2014-03-25

Semantic Technology Institute Innsbruck

STI INNSBTRUCK
Technikerstraße 21a
A – 6020 Innsbruck
Austria
http://www.sti-innsbruck.

Contents

1. Introduction

This document presents the implementation of the content and ontology mapping for Tourismusverband (TVb) Innsbruck. The outputs of this implementation are machine processable objects of all concepts in the content.

The main functions of the object models are:

1) As fact models to drive the rules. A Rule Based System contains Facts and Rules, where the rules are constructed based on the facts by an expert. Typically, an integrated user interface is used for rule creation (i.e. Guvnor[1]) to avoid any mistakes. In the case of Guvnor, the fact models can be defined through the declarative model of Guvnor itself or upload a Java archive in the format of Plain Old Java Object[2]. The Guvnor will handle the fact models as assets for rule creation.
2) As fact models to define the application's knowledge. In case of Drools[3], the knowledge base can be modified by inserting, updating and retrieving facts to or from its working memory.
3) As internal data representation for the Weaver. The Weaver requires an internal data model to represent the incoming inputs (i.e. annotated HTML sources), to perform some internal processing (i.e. content transformation) before deliver it to the selected output channels.

This implementation is based on the activity previously conducted to mapping the content to ontology. The Schema.org[4] has been chosen as the primary vocabulary for the mapping, therefore the object models will use it as much as possible.

2. Object Model

In this section, the dependency relationships among the main objects will be explained first and continued with the super and sub-objects relationships.

There are 10 main objects:

1. Hotel
2. FoodAndDrinkEstablishments
3. Event
4. TravelAction
5. PlaceOfInterest
6. BlogPosting
7. Offer

[1] See: http://www.jboss.org/drools/drools-guvnor.html
[2] See: http://docs.jboss.org/drools/release/5.2.0.Final/drools-guvnor-docs/html/ch05.html
[3] See: https://www.jboss.org/drools/
[4] See: http://schema.org

8. Place
9. PostalAddress
10. Country

The names for all main objects are borrowed from the Schema.org except FoodAndDrinkEstablishments and PlaceOfInterests. Those new objects are introduced as super objects for various kinds of items in both categories. The relationships among those objects are shown at Figure 1.

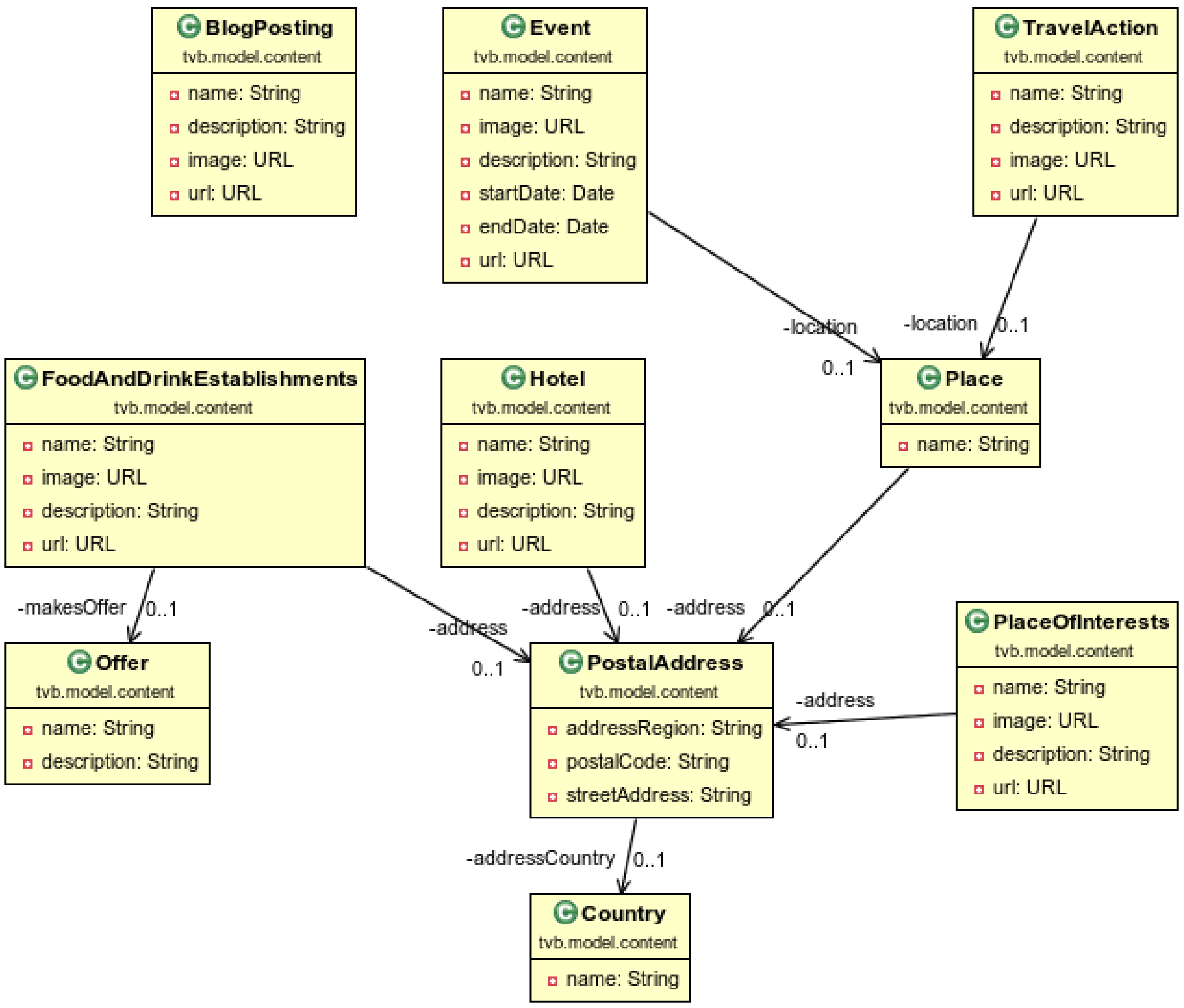

Figure 1 Dependency relationships among main objects

As shown at Figure 1, all objects have relation to other objects except for the BlogPosting. Event and TravelAction are related to Place through property location. Hotels, Place, PlaceOfInterests, and FoodAndDrinkEstablishments are related to object PostalAddress through property address.

PostalAddress is relating to Country through property addressCountry and the FoodAndDrinkEstablishments has second relation to object Offer through property makesOffer.

It is worthy to mention that there are two types of objects to represent a place, by Address (as shown in Hotel to PostalAddress relationship) or by Location (as shown in Event to Place relationship). The Location is representing a more broad concept of place (an object Location can be named) while the Address/PostalAddress object is representing a more specific/detail place.

FoodAndDrinkEstablishments is a super object contains various sub-objects in the category of Food and Drink Establishments. From previous analysis, we have identified several sub-objects: Bakery, Bar, Pub, Cafe, FastFoodRestaurant, IceCreamShop, Restaurant, and Winery, which are represented in the object models as shown at Figure 2.

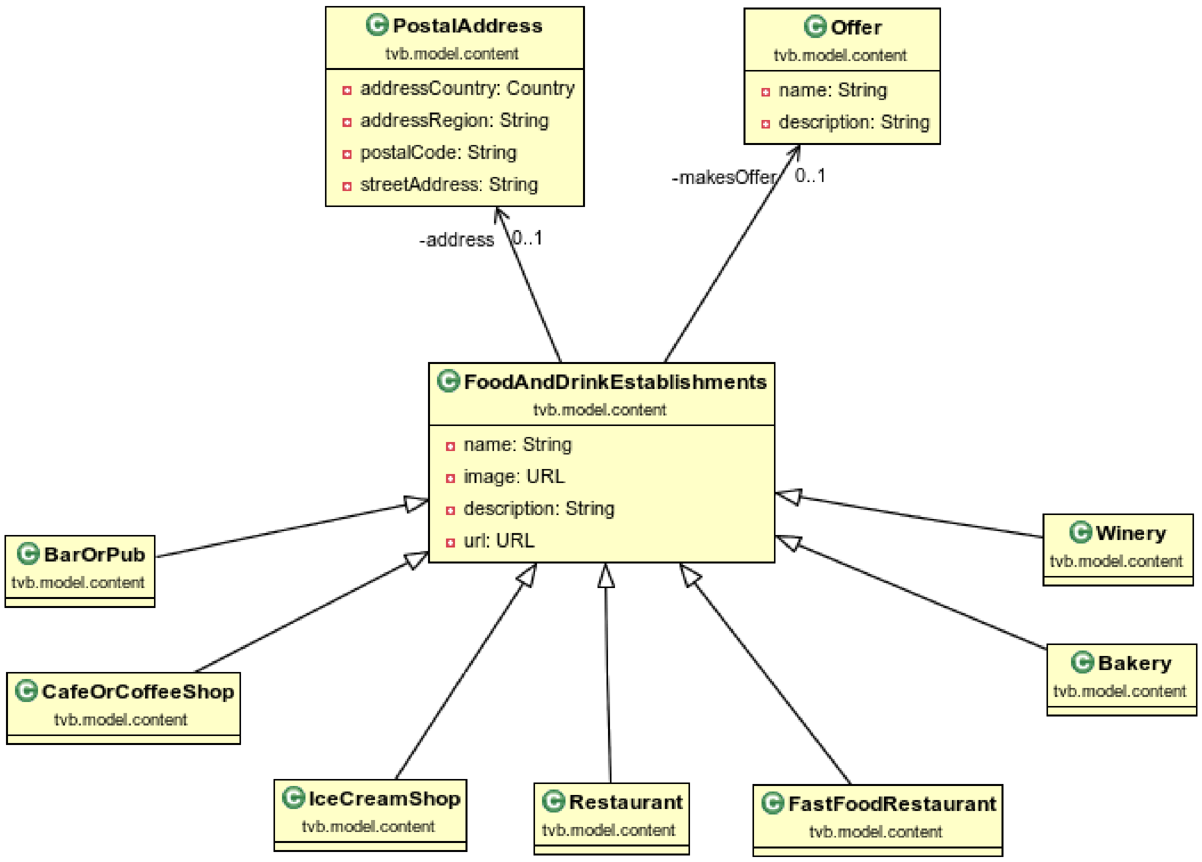

Figure 2 Food and Drink Establishments sub-objects relationships

At shown at Figure 2, there are seven sub-objects that are using Schema.org vocabularies. Those sub-objects will inherit all properties from their super object (FoodAndDrinkEstablishments) including the dependencies to the object Offer and PostalAddress.

PlaceOfInterests is super object for interesting places for tourists. Various sub-objects have been identified as shown at figure 3.

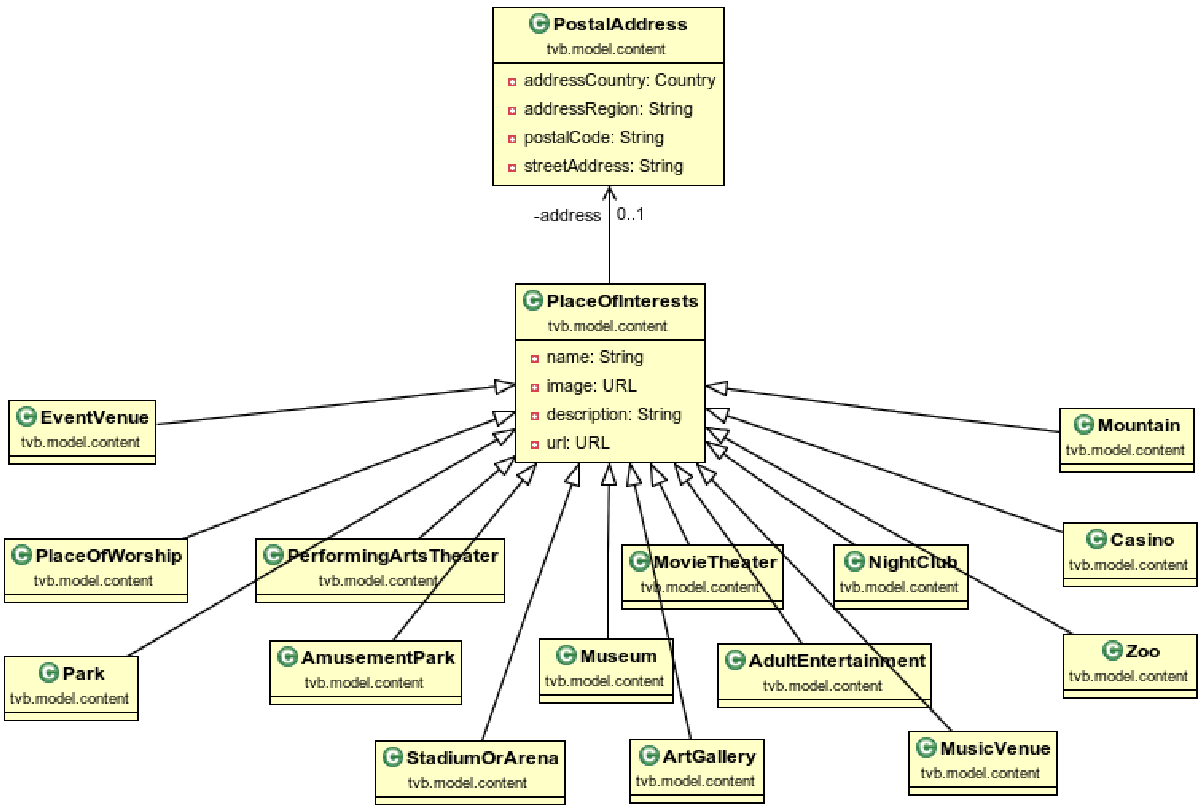

Figure 3 Place of Interests sub-objects relationships

As shown at Figure 3, the sub-objects (presented using the Schema.org vocabularies) are EventVenue, PlaceOfWorship, Park, PerformingArtsTheater, AmusementPark, StadiumOrArena, Museum, MovieTheater, AdultEntertainment, NightClub, MusicVenue, ArtGallery, Casino, Zoo, Mountain. Each sub object will inherits the properties of the PlaceOfInterests including the relation to object PostalAddress.

3. Implementation

In this section, we will explain current implementation including the technologies used and problems encountered. A possible solution is presented in the next section.

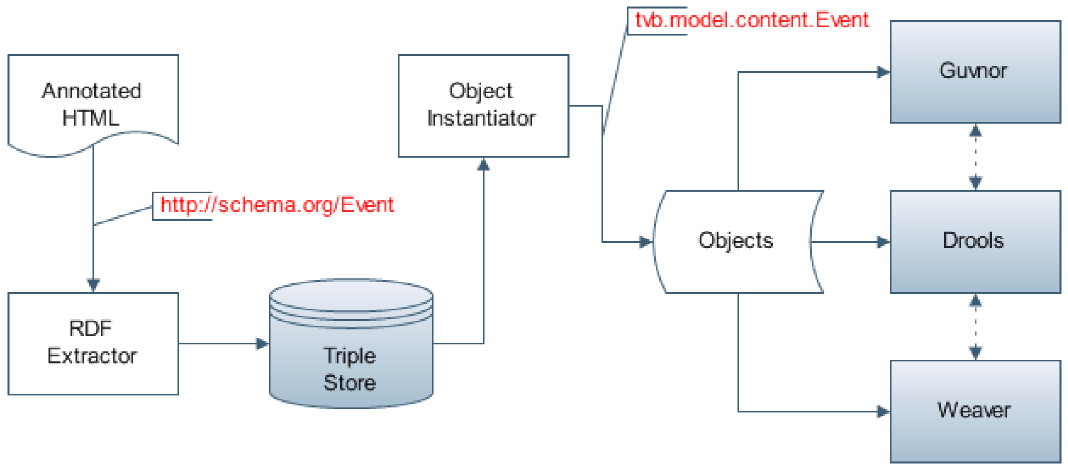

Figure 4 Mapping implementation

Figure 4 shows the current implementation. An RDF Extractor consumes the incoming HTML inputs (annotated with Schema.org) and produces triples, which are stored in a triple store. Based on the stored triples, relevant objects are instantiated. These objects are used by the rule editor (Guvnor), the rule execution engine (Drools) as well as the Weaver.

The RDF Extractor can be implemented with various existing libraries (i.e. Apache Any23[5]) easily. However, object instantiation turned out to be problematic. There are challenges due to the properties of the RDF itself, mainly because:

1. Complex associations which span across multiple blank nodes are hard to resolve
2. A tree-like structure of concepts and sub-concepts is equally hard to instantiate

To overcome those problems, in this initial implemention we are using several assumptions:

1. Complex properties that are associated with other objects are ignored
2. The object structure is flat

We are currently looking into a solution to solve the problem using an existing framework called RDFBeans[6], which is providing a mechanism for mapping an object-oriented domain model to RDF resource descriptions.

If this solution turns out to solve our problems completely, the assumptions we currently make are nullified. The instantiated objects would represent the RDF graph in its entirety.

[5] Apache Any23, https://any23.apache.org
[6] RDFBeans Framework, http://rdfbeans.sourceforge.net

4. Discussion

There are several points we would like to point out for discussion:

1) As stated in the beginning of this document, we would like to use Schema.org vocabularies as much as possible. It is clear that this implementation has adopted a few parts of Schema.org but dismissed their original structures. For example, in Schema.org, the Hotel is structured as Thing.Organization.LocalBusiness.LodgingBusiness.Hotel. In this implementation, those structures have been dismissed for simplicity reason. We believe that in TVb Innsbruck case, keeping the original structures will not contribute to the dissemination process. More than that, since this implementation is using the same vocabularies, we can track back to their original structures easily.
2) A new idea has emerged to solve the encountered problems in this implementation. The idea is to use an RDF reasoner instead of a rule engine. In this case, the reasoning will take place in the internal triple store, making the instantiation of objects obsolete.