

Uniwersytet Ekonomiczny w Katowicach

Kierunek: Informatyka

Przedmiot: Technologie semantyczne

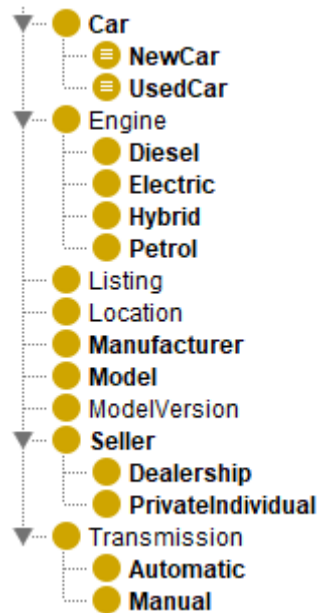
2024/2025

# **Dokumentacja projektu cars\_ontology**

Jan Bielski  
Paweł Krawczyk

## Opis ontologii

Ontologia opisuje proces sprzedaży samochodu. Jest ona luźnie inspirowana popularnymi stronami internetowymi z ogłoszeniami samochodów na sprzedaż. W projekcie nie wykorzystano innych, istniejących już ontologii. Ontologia zapisana jest w formacie RDF/OWL.



Lista klas

Ontologia definiuje samochody oraz ich atrybuty, a także ogłoszenia i sprzedawców. Niektóre klasy posiadają podklasy, które odzwierciedlają szczególne przypadki dla danych klas.

Description: NewCar

Equivalent To

+

● Car

and (hasMileage value 0)

SubClass Of

+

General class axioms

+

SubClass Of (Anonymous Ancestor)

● (hasEngine some Engine)

and (hasModelVersion some ModelVersion)

and (hasTransmission some Transmission)

Instances

+

Target for Key

+

Disjoint With

+

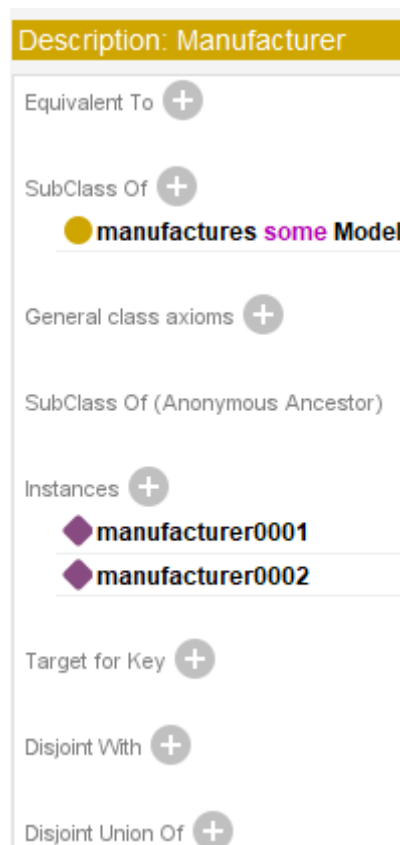
≡ UsedCar

Disjoint Union Of

+

Przykład klasy spełniającej warunek konieczny i wystarczający w ontologii (a także przykład klasy rozłącznej)

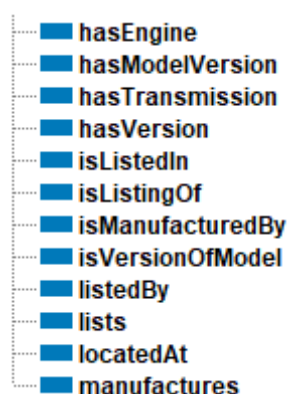
Na zrzucie ekranu widać, jak została zdefiniowana klasa NewCar oznaczająca nowe auto. Warunek konieczny i wystarczający jest taki, że obiekt klasy NewCar jest jednocześnie obiektem klasy Car i musi mieć wartość relacji hasMileage równą 0. Oznacza to, że samochód ma przebieg równy 0. Ponadto klasa NewCar jest klasą rozłączną z klasą UsedCar oznaczającą auto używane.



Przykład klasy spełniającej warunek konieczny

Na zrzucie ekranu widać, jak została zdefiniowana klasa Manufacturer oznaczająca producenta. Warunek konieczny i wystarczający jest taki, że obiekt klasy Manufacturer ma relację manufactures some Model, co oznacza, że producent produkuje jakiś model (samochodu).

Również niektóre klasy mają zdefiniowane warunki konieczne oraz warunki konieczne i wystarczające.



Lista relacji między klasami

Relacje w tej ontologii dotyczy głównie samochodów. Choć w większości trójek nie pojawia się bezpośrednio klasa „Car” to jednak wszystkie powiązania prowadzą do aut.

Characteristics: lists

☐ Functional
 ☒ Inverse functional
 ☐ Transitive
 ☐ Symmetric
 ☒ Asymmetric
 ☐ Reflexive
 ☒ Irreflexive

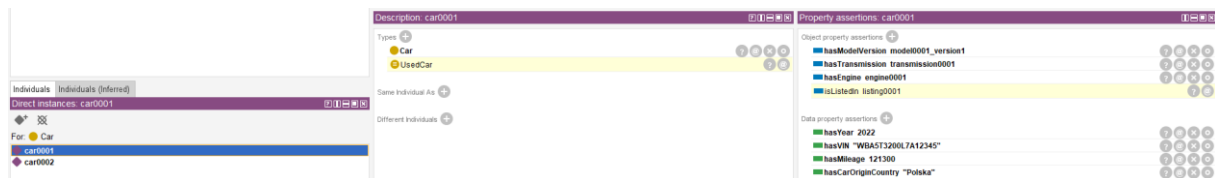
Przykład zastosowanych własności relacji

Relacja lists (tutaj oznacza „wystawia ogłoszenie”) jest odwrotnie funkcjonalna (takie jest założenie tej ontologii), asymetryczna i przeciwzwrotna. Wszystkie relacje ontologii są asymetryczne i przeciwzwrotne, ale nie wszystkie są odwrotnie funkcjonalne, ponieważ mniej więcej połowa relacji jest funkcjonalna.

- ☒ hasAddress
- ☒ hasCarOriginCountry
- ☒ hasCity
- ☒ hasCountry
- ☒ hasDescription
- ☒ hasEmailAddress
- ☒ hasEngineName
- ☒ hasExpiryDate
- ☒ hasHorsePower
- ☒ hasManufacturerCountryOfOrigin
- ☒ hasManufacturerName
- ☒ hasMileage
- ☒ hasModelName
- ☒ hasModelVersionName
- ☒ hasNumberOfGears
- ☒ hasPhoneNumber
- ☒ hasPostedDate
- ☒ hasPrice
- ☒ hasSellerName
- ☒ hasTaxId
- ☒ hasTitle
- ☒ hasTransmissionName
- ☒ hasType
- ☒ hasVIN
- ☒ hasVolume
- ☒ hasYear
- ☒ hasZIPCode
- ☒ producedFrom
- ☒ producedUntill

Lista własności klas

Własności klas są bardziej zróżnicowane od relacji i pozwalają na określenie szczegółów opisujących sprzedawanych samochodów, ogłoszeń i sprzedających. Zdecydowana większość własności jest funkcjonalna.



Przykładowa instancja klasy Car

Oprócz tego, jak wygląda przykład instancji auta, zrzut ekranu pokazuje, że reasoner znalazł relacje, które wynikają z ontologii, a które nie zostały jawnie określone. Relacje te zostały zdefiniowane poprawnie.

## Opis aplikacji

Ta aplikacja służy do interakcji z bazą danych zawierającą oferty sprzedaży samochodów, opartą na ontologii. Umożliwia użytkownikom filtrowanie, pobieranie i odpytywanie ogłoszeń sprzedaży aut i powiązanych metadanych za pomocą zapytań SPARQL wykonywanych na serwerze Apache Jena Fuseki. Aplikacja zapewnia interfejsy API dla różnych funkcji, takich jak pobieranie ofert samochodów, filtrowanie danych według atrybutów i określanie przedziałów cenowych.

Podstawowym język programowania zastosowanym w kodzie aplikacji jest Python. Aplikacja wykorzystuje framework Django do zarządzania ustawieniami, w szczególności do przechowywania endpointu Fuseki. Apache Jena Fuseki jest serwerem hostującym ontologię i umożliwiający wysyłanie zapytań w języku SPARQL. SPARQL to język zapytań pozwalający na interakcję z ontologią. Ważną funkcję pełni biblioteka SPARQLWrapper, która służy do wykonywania zapytań SPARQL na serwerze Fuseki. Zapewnia metody konstruowania zapytań i formatowania wyników.

Zapytania są konstruowane dynamicznie przy użyciu filtrów. Prefiksy ontologii są dodawane automatycznie. Zapytania są wysyłane do endpointu Fuseki przy użyciu biblioteki SPARQLWrapper. Wyniki są pobierane w formacie JSON, a następnie przetwarzane w celu wyodrębnienia odpowiednich pól. Dane są formatowane do słowników lub list Pythona w celu łatwego wykorzystania.

Opis kluczowych funkcji:

### 1. execute\_sparql\_query

```
def execute_sparql_query(query):
    """
    Helper function to execute a SPARQL query against the Fuseki server.
    """
    sparql = SPARQLWrapper(settings.FUSEKI_SPARQL_ENDPOINT)
    prefix = """PREFIX : <http://www.semanticweb.org/janbi/ontologies/2025/0/cars-ontology/>\n
               PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>\n
               PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>"""
    query = prefix + "\n" + query
    sparql.setQuery(query)
    sparql.setReturnFormat(JSON)
    return sparql.query().convert()
```

Jest to funkcja pomocnicza, która wykonuje zapytania SPARQL na serwerze Fuseki, dodaje prefiksy ontologii do zapytania oraz zwraca wyniki zapytania w formacie JSON.

```
query = f"""
    SELECT ?title ?modelVersionName ?modelName ?manufacturerName ?postedDate ?expiryDate ?description ?mileage ?price
      (STRAFTER(STR(?transmissionType), "cars-ontology/") AS ?transmissionTypeName) ?transmissionName ?numberOfGears
      (STRAFTER(STR(?engineType), "cars-ontology/") AS ?engineTypeName) ?engineName ?volume ?horsePower ?countryOfOrigin
      ?vin ?year ?address ?city ?country ?zipCode
      (STRAFTER(STR(?sellerType), "cars-ontology/") AS ?sellerTypeName) ?sellerName ?taxId ?modelVersionType ?carState WHERE {{
?listing a :Listing ;
    :hasTitle ?title ;
    :isListingOf ?car ;
    :hasPostedDate ?postedDate ;
    :hasExpiryDate ?expiryDate ;
    :hasDescription ?description ;
    :hasPrice ?price ;
    :locatedAt ?location ;
    :listedBy ?seller .
?car :hasModelVersion ?modelVersion ;
    :hasMileage ?mileage ;
    :hasTransmission ?transmission ;
    :hasEngine ?engine ;
    :hasVIN ?vin ;
    :hasYear ?year ;
    :hasCarOriginCountry ?countryOfOrigin .
?modelVersion :isVersionOfModel ?model ;
    :hasModelVersionName ?modelVersionName ;
    :hasType ?modelVersionType .
?model :hasModelName ?modelName ;
    :isManufacturedBy ?manufacturer .
?manufacturer :hasManufacturerName ?manufacturerName .
?transmission rdf:type ?transmissionType ;
    :hasTransmissionName ?transmissionName ;
    :hasNumberOfGears ?numberOfGears .
?transmissionType rdfs:subClassOf :Transmission .
?engine rdf:type ?engineType ;
    :hasEngineName ?engineName ;
    :hasVolume ?volume ;
    :hasHorsePower ?horsePower .
?engineType rdfs:subClassOf :Engine .
?location :hasAddress ?address ;
    :hasCity ?city ;
    :hasCountry ?country ;
    :hasZIPCode ?zipCode .
?seller rdf:type ?sellerType ;
    :hasSellerName ?sellerName .
?sellerType rdfs:subClassOf :Seller .
OPTIONAL {{
    ?seller a :Dealership ;
    :hasTaxId ?taxId .
}}
    BIND(IF(EXISTS {{ ?car a :NewCar }}, "New", "Used") AS ?carState)
    {filter_string}
}}
"""
```

Zmienna query, która funkcja przyjmuje jako argument

Za zmienną query kryje się zapytanie SPARQL. Przykładowe zapytanie (wykorzystywane przez funkcję `fetch_filtered_listings`), zwracające szczegółowe informacje o ogłoszeniach sprzedaży samochodów, przedstawiono powyżej. Wyniki zapytania prezentują pełen obraz dostępnych ogłoszeń z podziałem na szczegóły samochodu, jego specyfikacji technicznej, lokalizacji i informacji o sprzedawcy.

## 2. `fetch_filtered_listings`

Ta funkcja pobiera oferty samochodów spełniające określone kryteria filtrowania.

Car Listings

Filters

Type

All

Brand

All

Model

All

Engine

All

Transmission

All

Min Price

2000 PLN

Max Price

120000 PLN

Apply Filters

Piękne BMW G21

BMW | Seria 3 | G21

Location: Feliksa Bocheńskiego 66, Katowice 40-847, Polska

Na sprzedaż przepiękne BMW, nie bite, starszy człowiek jeździł tylko na zakupy i do kościoła.

Seller Information

Type: Dealership

Name: BMW Gazda Group

Tax ID: 6380012661

Engine Information

Type: Petrol

Name: B48 R4 Turbo

Volume: 2.0 L

Horse Power: 181 HP

Car Information

Type: Sedan

Year: 2022

VIN: WBA5T3200L7A12345

Mileage: 121300 km

Country of Origin: Polska

Transmission Information

Type: Automatic

Name: Steptronic

Number of Gears: 8

Posted: 2024-01-30T00:00:00

Expires: 2024-01-30T00:00:00

PRICE: 120000 PLN

Znakomity peugeotik szuka nowego właściciela

Peugeot | 306 | PH1

Location: Jasna 23, Chorzów 41-506, Polska

Przepiękny pojazd, takich już nie robią. Pali, skręca i hamuje.

Seller Information

Type: PrivateIndividual

Name: Jan Nowak

Engine Information

Type: Diesel

Name: PSA DW10 HDi

Volume: 2.0 L

Horse Power: 90 HP

Car Information

Type: Hatchback

Year: 1998

VIN: VF3LC9HP0HS123456

Mileage: 451000 km

Country of Origin: Polska

Transmission Information

Type: Manual

Name: 20TB51

Number of Gears: 5

Posted: 2024-01-02T00:00:00

Expires: 2024-02-02T00:00:00

PRICE: 2000 PLN

Podgląd aplikacji z domyślnymi filtrami

Domyślnie filtry ustawione są tak, by pokazywać wszystkie dostępne ogłoszenia, niezależnie od ich charakterystyki. Funkcja pozwala na dynamiczne konstruowanie zapytań SPARQL w celu zastosowania określonych filtrów.

Car Listings

Filters

Type

Hatchback

Brand

All

Model

All

Engine

All

Transmission

All

Min Price

2000 PLN

Max Price

120000 PLN

Apply Filters

Znakomity peugeotik szuka nowego właściciela

Peugeot | 306 | PH1

Location: Jasna 23, Chorzów 41-506, Polska

Przepiękny pojazd, takich już nie robią. Pali, skręca i hamuje.

Seller Information

Type: PrivateIndividual

Name: Jan Nowak

Engine Information

Type: Diesel

Name: PSA DW10 HDi

Volume: 2.0 L

Horse Power: 90 HP

Car Information

Type: Hatchback

Year: 1998

VIN: VF3LC9HP0HS123456

Mileage: 451000 km

Country of Origin: Polska

Transmission Information

Type: Manual

Name: 20TB51

Number of Gears: 5

Posted: 2024-01-02T00:00:00

Expires: 2024-02-02T00:00:00

PRICE: 2000 PLN

Przykład wykorzystania funkcji w praktyce - filtrowanie wg typu samochodu

Funkcja zwraca szczegółowe informacje o pasujących ofertach, co pozwala funkcji home na poprawne wyświetlenie zastosowanego filtrowania.



3. `fetch_all_listings`, `fetch_all_manufacturers`, `fetch_all_models`,  
`fetch_all_transmission_types`, `fetch_all_engine_types`,  
`fetch_all_model_version_types`

Te funkcje zostały przypisane do tego samego punktu, ponieważ mają podobne zadanie – zwracają wszystkie zdefiniowane w ontologii kolejno: ogłoszenia, producentów, modele, rodzaje skrzyni biegów, rodzaje silników i wersje modeli. Te z kolei są następnie wykorzystywane przez funkcję `home`.

#### 4. `get_price_range`

Funkcja zwraca minimalną i maksymalną cenę znajdujące się w ogłoszenia w ontologii. Pozwala to funkcji `home` na utworzenie suwaków, którymi można manipulować widełkami cenowymi.

The screenshot displays a web interface for car listings. On the left, there is a 'Filters' sidebar with dropdown menus for 'Type', 'Brand', 'Model', 'Engine', and 'Transmission', all currently set to 'All'. Below these are price range sliders for 'Min Price' (set to 10000 PLN) and 'Max Price' (set to 120000 PLN). An 'Apply Filters' button is at the bottom of the sidebar. The main content area shows a detailed listing for a 'Piękne BMW G21'. It includes the car's location, a description, and several information sections: 'Seller Information' (Dealership, BMW Gazda Group), 'Engine Information' (Petrol, B48 R4 Turbo, 2.0 L, 181 HP), 'Car Information' (Sedan, 2022, VIN: WBA5T3200L7A12345, 121300 km), and 'Transmission Information' (Automatic, Steptronic, 8 gears). The price is listed as 'PRICE: 120000 PLN'.

Przykład wykorzystania funkcji w praktyce - filtrowanie wg ceny

W tym przykładzie ustawienie minimalnej ceny na poziomie 10 000 zł powoduje, że wyświetla się tylko jeden samochód, ponieważ druga oferta ma cenę 2 000 zł.

#### 5. `home`

Ta funkcja powiązana z Django służy za backend dla strony głównej aplikacji. Przetwarza dane modyfikowane przez użytkownika, pobiera odpowiednie dane i przekazuje wyniki do szablonu w celu ich wyświetlenia. Użytkownik podaje filtry za pomocą parametrów zapytania adresu URL. Funkcja `home` przetwarza te filtry, pobierając odpowiednie listy samochodów z bazy danych za pomocą zapytań SPARQL. Pobiera również listy dostępnych atrybutów, które mogą być wykorzystane w opcjach filtrowania. Na koniec widok wyświetla szablon `home.html`, przekazując wszystkie niezbędne dane (np. oferty samochodów, opcje filtrowania, zakresy cen) do szablonu w celu wyświetlenia.

## Car Listings

### Filters

Type

Hatchback



Brand

BMW



Model

All



Engine

All



Transmission

All



Min Price



2000 PLN

Max Price



120000 PLN

Apply Filters

No results found

Przykład użycia aplikacji - zastosowanie takich filtrów, które nie pokrywają się z żadnym ogłoszeniem w ontologii