Abstract

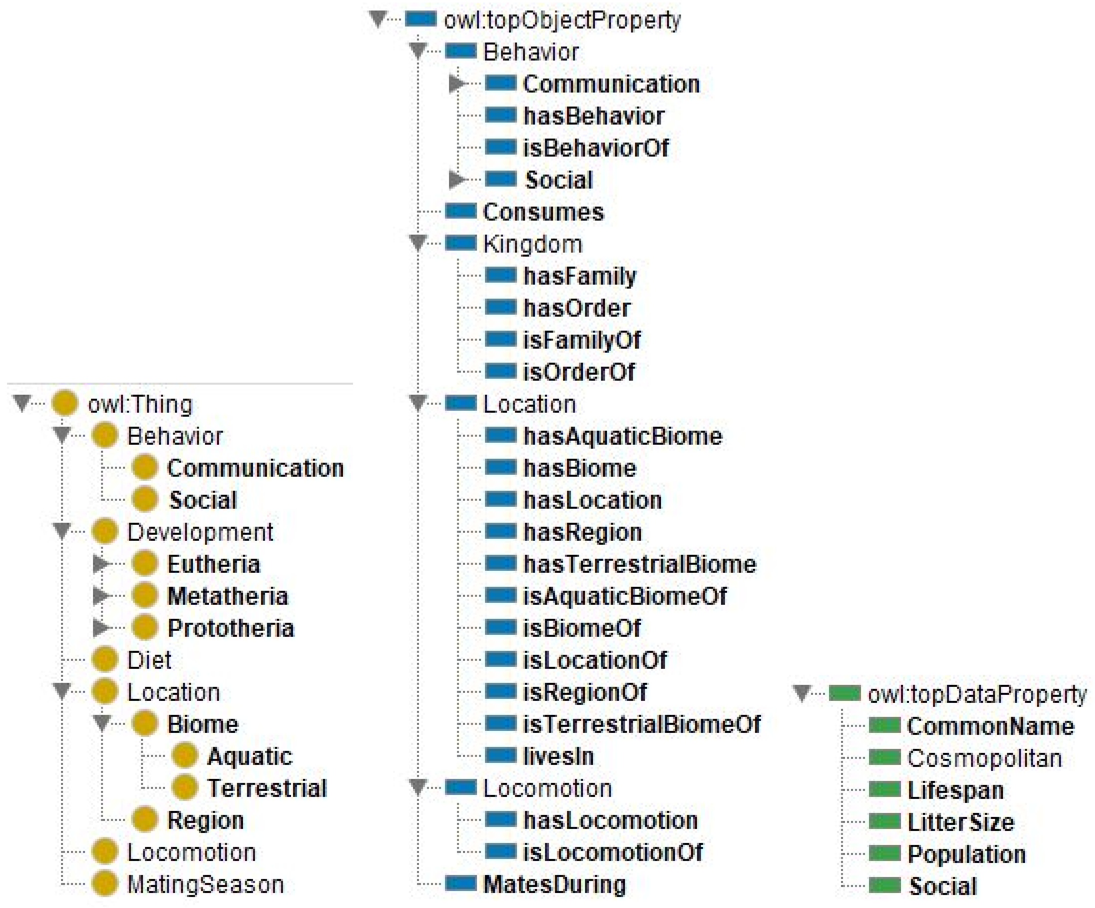
Mammal information is widely available on the internet. However, there are limited resources which consolidate such information in one place, let alone one which is easily accessible. This database is designed for educational purposes only; facilitating information gathering regarding specific mammal species and/or species that fall into specific generalizations. All information obtained and used in the database are from publically accessible and free-to-use sources. The database was created in Protege and interfaced with using the Flask web service. The interfacing went together seamlessly and any issues or questions regarding implementing certain functionality was answered with Google-ing. Google-ing is very effective.

Data Processing

As per the project requirements, this dataset features several classes and properties with a sizable amount of instances to accompany them. In the early stages, there were entities which encompassed more behavioral traits of mammals, however these were later removed due to challenges in obtaining information regarding such entities. The classes included in the final project are based on my own interest regarding such features / facts. The database was populated by hand in Protege, with information being supplied from various sites on the Internet. All the sites were publically accessible and unlicensed. All data used in populating the mammal database was obtained and entered manually in April, 2020. Therefore, the database reflects all relevant data up until that date.

RDF Design

The final RDF design features six main classes: Behavior, Development, Diet, Location, Locomotion, and Mating season. Development is subdivided into the three Mammal groups and further into each order which makes them up. There is at least one instance for every "order" of Mammal. There are various object properties, notably ones which connect Behaviors, Diet, Location, and Mating Season with Development. This allows each instance of a Development species to have the above characteristics linked. Properties not encompassing their own class are data properties. The database sports five main data properties: CommonName, Lifespan, LitterSize, Population, and Social. For basic overview reference the images located below. For a more detailed design, locate the original OWL file in the static folder and open it using Protege.



Database Creation

The RDF API used in this particular project is RDFLib. This project takes advantage of the library's RDF parser and graph interface to construct a graph that can be manipulated during Flask server runtime. This API functionality is not limited to handling SPARQL queries to manage the database, hosting several native graph methods to easily add, query, remove, and update triples. One neat feature, coined as the "purge removal," can delete all triples which relate to a specific subject. This can be very powerful and removes the necessity of deleting each triple individually. Furthermore, searching can be easily accomplished through filling out the subject predicate and object boxes and the API returns all triples which correspond to the search fields entered. Specifics pertaining to how this API was integrated within the website itself can be found in the following section.

Web Application

The website itself is served over Flask, a python microframework for web applications. Althouthough it has no database abstraction layer, there are several Python libraries that implement such functionality. Following the Flask framework layout, the HTML pages are stored in a templates folder with the CSS files, JavaScript file, favicon, and OWL file stored in a static folder. A python folder located in the main directory drives the server and hosts the HTML pages. This organization structure not only makes separation of various files easier to manage and access but reinforces good coding practices of code separation. The web application is divided into four HTML files. The main file, layout.html, merely acts as a template and imports the HTML from the other respective three files. As importing separate HTML files is not natively supported through javascript or HTML, I wrote my own function using HTML objects and appending them to a div element before removal. The tab effect seen on the data management section is handled almost purely through css and buttons. Each button corresponds to HTML elements which are hidden/shown every time a tab button is clicked.

The RDF API used in this particular project is RDFLib. This project takes advantage of the library's RDF parser and graph interface to construct a graph that can be manipulated during Flask server runtime. URI's are created and saved in a dictionary to allow users to use prefix notation when constructing triples. User input handled using HTML forms. Instead of directing to a particular php file, the website is redirected to a different URL which handles the post method. The routing handled by the dev.py file obtains the information from the POST method and feeds it into the same layout.html file as the home-page. The reason behind this design was to provide a structure which allowed the use of Python and it's corresponding libraries. Before the result is passed and rendered in the layout.html file, it interacts with the RDF graph stored in memory. The results are displayed using HTML Tables and translated into their prefix form before they are made visible.

As most users are unfamiliar with the command line and how to run the Flask server, I created a batch file which when double-clicked or run, starts up the server and loads the graph. The user still has to navigate to http://127.0.0.1:5000/ IP address on their own.

Conclusion

Mammals are fascinating creatures that vary drastically from each other. This database sets out to capture said differences between various orders by providing a go-to educational and easily accessible database. The database was composed in Protege and served over Flask. Although the structure of the RDF was designed solely by me, the data used to populate it was obtained off the Internet from free and unlicenced sources. The project uses minimal JavaScript but relies heavily on CSS for the layout and format of the entire website. Custom HTML importing was implemented throughout this project to increase code separation and modularity. This prevented a single file from reaching well over 1,000 lines of code on it's own. Imagine the nightmare of singling out a particular statement! The database is still relatively small, with only 15 major classes and only 5 data properties, totaling 1,252 triples without additions or subtractions. It does, however, include an instance of a species for every order of Mammal currently recognized where its constituent members are not all extinct.