Designing a faceted search for solar data

Ian Ruotsala

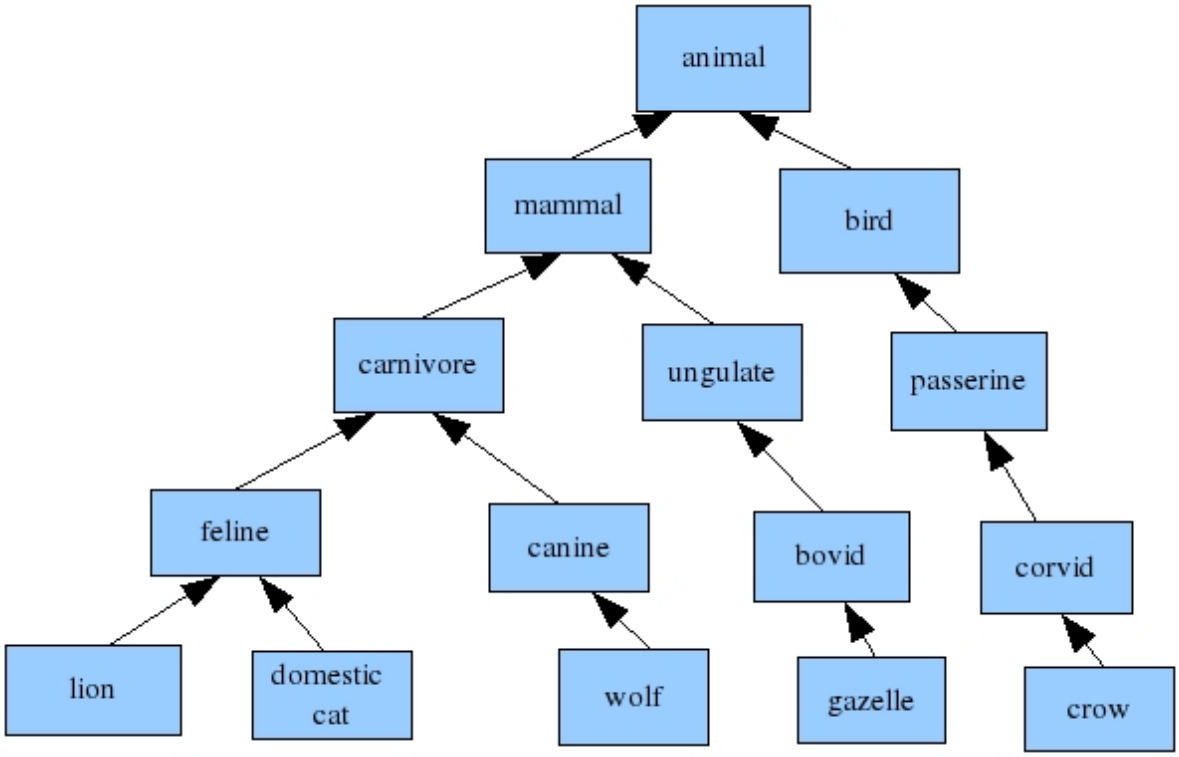
Abstract

In library, information and computer science, an ontology is a network of relationships. A well-known subset of an ontology is a "taxonomy," which is an ontology consisting of "is a" relationships. Biological taxonomies such as "a human is a primate is a mammal is an animal" provide a concrete example. While taxonomies have often been used by software developers as a basis for categorizing a set of items, more general ontologies can provide a richer description of relationships within a set.
Faceted searches provide a way to navigate through instances of a more general ontology and have been increasingly used by Internet merchants such as Amazon to provide intuitive ways of searching and browsing through their products. For this project, an implementation of faceted search was attempted for solar physics data. Using an Internet browser as a platform, a JavaScript program was to be developed in which a data base of solar events (e.g. flares, sunspots) and observations (e.g. which instrument was used, which astronomer cataloged the event) would be queried so that data could be accessed in a dynamic, intuitive way.
Keywords:
faceted search, ontology, search, JavaScript, software development, web application, helioinformatics, solar physics

Background: ontologies and faceted search

Taxonomies are constrained ontologies

Imagine one were to enumerate a set of animal species. One could classify the species according to their types, e.g.:



This is an example of a taxonomy, a network of "is a" relationships.

This is indeed useful, yet one could imagine many other sorts of relationships among species than type. One could describe "lion preys upon gazelle" or "hyena competes with lion". This more thorough network of relationships is an example of an ontology.

Faceted search is implemented by Internet merchants

It is common for Internet merchants to categorize their merchandise in an ontology, so that a faceted search may be implemented. Whereas traditional type classification encouraged hierarchal browsing according to a single relationship ("is a"), faceted search is a way of browsing through an ontology in lateral ways, and using many different relationships.

Helioinformatics

With increasingly cheap computer storage and processing, more massive amounts of data are generated by scientific studies. In solar physics, the field known as "helioinformatics" uses various tools of informations technology, including ontologies, to better get a handle on data.

Ontology of solar data

During the summer of 2009, I interned at the Lockheed Martin Solar & Astrophysics Laboratory (LMSAL). LMSAL has a database of solar data which is accessible via a web API (application programming interface). My task over that summer and into this school year has been to implement a faceted search for their solar data using that API.

Implementation challenges and achievements

Background: description of current system

LMSAL maintains a database of solar events and observations, the Heliophysics Event Registry (HER). The HER can be queried from a web API, returning a JSON file that contains information on events and observations within specified parameters, as well as the relationships of those events and observations to other events and observations. The JSON file format is a data-interchange format (a la XML), as well as a subset of JavaScript, making it especially pliable to that language. My faceted search was to be incorporated into their present search interface at http://www.lmsal.com/helio-informatics/hpkb/

Initial approach

The initial idea was to have the search be structured in a tree-like fashion, with a description of the currently searched set at the root, and, for leaf nodes, a description that was describable by the more limited web API. To mediate between the leaves and the root, various intermediary nodes would be used.

Changing approach

I think that I did not fully understand how faceted searching worked when I decided on the initial approach. Fortunately, I decided on a better approach; unfortunately, it was not until mid-way through my project. With my new approach, the user would be presented with every event and observation within a timespan. The user could then narrow down by selecting for specific features.
A challenge came up by way of the fact that the API only allows 200 events to be returned per query. My way around this was to have each of the about twenty different event types do a query involving just themselves; so, e.g., the "solar flare" type would only search for solar flares within a time span, the sunspot only sunspots, and so on. Once returned, the various events would be

stored together so that events with different types but same properties (spots and flares both have grid locations, for example) could be found in the appropriate search.

Obstacles to completion

There were a variety of hindrances that kept this project behind schedule. Most significant among them, I underestimated the steepness of JavaScript's learning curve. JavaScript is described as having "Java-like syntax," and I foolishly thought that, since I could already program in Java, JavaScript would easily come to me. Despite the surface similarity of the two languages, they are, in fact, quite dissimilar: JavaScript is intended to be run on web browsers, while Java is run on a JVM; contrary to Java, there are no classes in JavaScript, only instantiated objects; and so on.

Coda

Sadly, this project is not near completion as of writing time for this paper. It was, however, a valuable learning experience. My plan is to continue working on this project even after the end of the school quarter with the hope that a modicum of extra effort, I can complete this.

Thanks

I would like to thank the many people whose insight, assistance and understanding brought this project along, including:

EJ Zita (The Evergreen State College) for providing the opportunity and preparation for the solar physics internship

Neal Hurlburt (Lockheed Martin Solar Astrophysics Lab) for his guidance on the direction of this project

Neal Nelson, *Sherri Shulman* and *Richard Weiss* (The Evergreen State College), my professors, for helping me mature as a programmer and student of computer science.