**Master’s Project Final Report**

***Enhancements for Ontology-Supported Web Search***

Project Report submitted to

The Department of Computer Science

New Jersey Institute of Technology

In Partial fulfillment of the requirements of the Degree of Master of Science in Computer Science

Submitted by Yuwen Sun

**NJIT ID: 218-46-485**

[**ys89@njit.edu**](mailto:ys89@njit.eduail)

APPROVALS

Proposal Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Agree to Advice: \_\_**Prof. James Geller**\_\_\_\_\_\_\_\_\_\_\_\_

(Project Advisor)

Date Submitted: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Approved by: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(MS in CS Committee)

Date Approved: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Acknowledgement**

My project ‘Enhancements for Ontology-Supported Web Search’ was a great learning experience. I would like to express my sincere gratitude to Dr. James Geller for providing me an opportunity to work under his guidance. Without his encouragement and guidance this project would not have been materialized.

I would also like to express my appreciation to Tian Tian, for her great support and suggestions in doing the project. Her advice to this project is invaluable.

The project wouldn’t have been complete without the support of Christopher Ochs who helped me many times on GWT development and David Perel from IST University Computing Systems, who helped me to successfully deploy this project on AFS.

**Abstract**

Keyword-based search engines face a complication when a search term is a homonym, that is, a word with multiple meanings. The ‘Ontology-Supported Web Search’ project aims at building domain specific ontologies with that information, categorizing the homonymous terms and finally providing the users with a friendly interface for analyzing the search terms entered and getting relevant Web sites to show the contribution of the ontology to Web search effectiveness. The suggestions of the homonyms are extracted from an ontology developed in previous research.

My concentration is on improving the user interface of the Ontology-Supported Web Search system. To help the user better understand and differentiate the homonymous terms, I redesigned the layout of OSWS system and implemented “Instant Search” and “Parallel Result” features. “Instant Search” is a new search enhancement that shows results as you type and “Parallel Results” means the search engine can display multiple search results for different suggestions within one single Web page. I also expanded the number of suggestions from 4 to 20 and created a paging mechanism for users to navigate suggestions from the first one to the 20th suggestion.

Table of Contents

Table of Contents 4

1. Introduction 5

1.1 What is an Ontology? 5

1.2 Efforts on Improving the Web Search Experience 7

1.3 Semantic Web Search 9

1.4 Role of Ontology in search for Web pages 10

2. Previous Work 12

2.1 Ontology-Supported Web Search 12

3. My Work 14

3.1 Update System from GWT 1.6 to GWT 2.1 14

3.2 Enhancement of User Interface 16

3.2.1 System Layout 16

3.2.2 Main Page Appearance 17

3.2.3 Search Result Page Appearance 17

3.2.4 Hover-Time Slider 18

3.3 Instant Search for Ontology-Supported Web Search 18

3.4 Parallel Result Display for Homonyms 21

3.5 Paging Support for Suggestions 22

4. Conclusions and Future Work 24

References 25

A. Appendix A: User Manual 28

A.1 Project Requirements 28

A.2 Recompile Jena 2.6.3 28

A.3 Setup Project in Eclipse 30

A.4 Compiling and Deploying project to AFS 32

B. Appendix B: Source Code 33

index.html 33

OntologySuggest.css 34

OSWS.gwt.xml 37

OntologySuggestService.java 37

OntologySuggestServiceAsync.java 38

OntologySuggestServiceImpl.java 38

OntologySuggestion.java 45

OSWS.java 46

Setting.java 47

Utils.java 48

OntologySuggestBox.java 48

SettingPanel.java 56

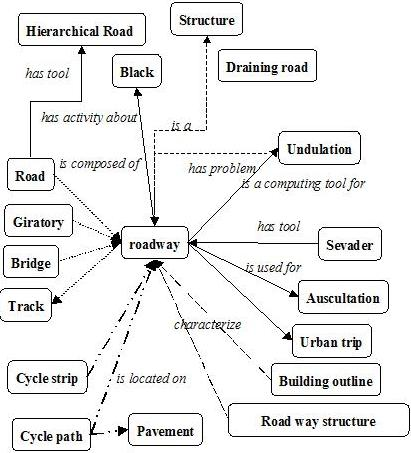
SuggestionPanel.java 58

# Introduction

## What is an Ontology?

In computer science and information science, ontology is a formal representation of knowledge as a set of concepts within a domain, and the relationships between those concepts. It is used to reason about the entities within that domain, and may be used to describe the domain [7]. In the words of Thomas Gruber ontology is an explicit specification of a conceptualization. A conceptualization is an abstract, simplified view of the world that we want to represent. If the specification medium is a formal language, the ontology defines a representational foundation [2].

Example: Figure 1 gives a pictorial representation of an ontology which consists of various classes and their corresponding relationships.



**Figure 1:** An example of an ontology [4]

Every node of this graph stands for a “concept” which is a unit that one can think about and corresponds to words or short phrases. Typically, a concept corresponds to a noun or noun phrase like house, man, car New York etc. but that is not an obligation [1].

The nodes of the ontology are connected by different kinds of links. The most important kind of link is called IS-A link. The nodes and IS-A links together form a Rooted Directed Acyclic Graph (Rooted DAG). Rooted means that there is one single “highest node” called the Root. All other nodes are connected by one IS-A link or a chain of several IS-A links to the Root. In our definition, IS-A links point upward. If an IS-A link points from a concept X to a concept Y that means that every real world thing that can be called an X also can be called a Y. In other words, every X IS-A Y. (Some people have IS-A-like links but pointing downwards). Examples: A car IS-A vehicle. A dog IS-A animal [1].

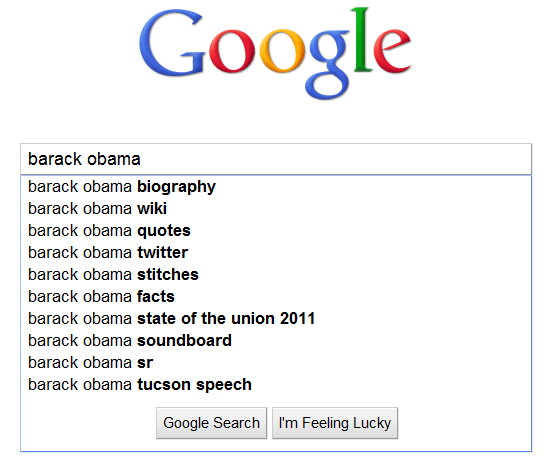
Acyclic means that if you start at one node and move away from it following an IS-A link, you can never return to this node, even if you follow many IS-A links. Most nodes also have other information attached. This information includes attributes, relationships and rules [1].

An ontology represents information in a form that can be used for some forms of reasoning that are at least partially similar to human reasoning. This includes inheritance reasoning, transitivity reasoning and classification. A concept may inherit information from several other concepts. This is called multiple inheritances. Transitivity reasoning corresponds to chaining of IS-A links. Classification means that if we know the attributes of a concept we can decide under which other concepts it belongs in the ontology.

## Efforts on Improving the Web Search Experience

In what is shaping up to be the “Century of the Web” a computer literate person with an information need is likely to eschew traditional sources of information such as libraries, yellow pages and newspapers and turn immediately to a Web search engine. Such information needs define the work sphere (“from where can I source this industrial part that I need”) as much as private life (“where is a nice, affordable restaurant near my home”) and everything in between (“I need a cheap flight for a job/private trip”). Thus, the quality of the search experience of a user has become of major importance. A user wants an answer, and she wants it **now,** and she wants it many times a day**.** Search engines are expected to provide correct results quickly, and with a minimal amount of user interaction.

To satisfy this expectation of an agreeable search experience, major efforts have gone into improving both the backends and frontends of common search engines. For example, Google has switched from making users type in complete search terms and hitting return (or clicking a button) to suggesting to the user what she is mostly likely to ask for. Such suggested completions [9] have also been introduced by other search engines. Figure 2 shows Google’s suggested completions for the query term “Barack Obama.” Google has access to the search terms entered by its millions of users, which makes it easy for them to propose crowd-based suggested completions.



**Figure 2:** Google’s suggested completions for search term “Barack Obama”

Changes to the backend are harder to discern for the user, but search results are often long lists of snippets referring to a few relevant links among many irrelevant results [10] [11]. Previous research has focused on refining the search terms and on filtering the results, to improve the precision of the returned snippets [10] [12] [13].

Search engines also suffer from three common problems in Natural Language Processing, the synonym problem, the homonym problem, and the wrong granularity problem. The synonym problem appears in the form that the user might send a different term to the search engine than what is contained in a document that would provide a relevant answer. Thus, a query term “43rd president of the US” might miss documents with George Bush, even though these two terms are synonymous.

The wrong granularity problem would appear when a user performs a search with a general or wide term, and a relevant document contains only a more specific or narrow term (or vice versa). Thus, a search for “government officials having been impeached” might not bring up President Clinton, who was indeed impeached.

The third problem in this category occurs when a search term is a homonym [6] [15] [16] (a term with multiple meanings or multiple referents) and the user does not know that. For example, when using the search term “President George Bush” without any further qualification, it might refer to George W. Bush or his father George H. W. Bush, the 43rd and 41st president of the United States, respectively. If the user wants information about the former, she would get results about both of them with this search term, which is an unintended and misleading result.

Thus, when using a search engine to find information about a homonymous term, two kinds of difficulties might arise. There might be an overwhelming number of responses about the more popular homonym, while the second homonym with a less popular meaning that she is really interested in is hidden in a snippet on a much later page of hits. This situation will occur for a user with an untypical information need. For example, the singer “Tony Bennett” appears to be more popular than the basketball coach of the same name, and/or more information about the singer is available in the public domain. Hence many more returned snippets are about the singer. In this situation, the user is at least aware that the results she is getting are not about the basketball coach that she has been looking for. At this point, she needs to wade through many hits for the wrong Tony Bennett or append terms to her query that will exclude the unwanted homonym and repeat the search.

If the user does not know that the search term is a homonym with two referents, she might not even notice if all snippets that appear on the first few pages of hits are to the "wrong" referent. For example, a user who types “Congo” into Google will see a large number of references to the Democratic Republic of the Congo and a few references to the Republic of the Congo. These two countries are located in Central Africa, bordering each other. The returned information about the two different countries is mingled together in the search results. Typing “Congo” into the Google search box leads to suggested completions such as river, bars, movie, war, etc. There is no hint to a user that there are two different countries called Congo in Africa.

## Semantic Web Search

Semantic search on the Web, which aims at enabling more intelligent Web searches, has become one of the hottest Semantic Web research topics [17]. Keyboard-based approaches have been studied by many researchers in the field to improve the search process [17]. For example, [18] improves the traditional search method by augmenting the search results with relevant data aggregated from the Semantic Web. Falcons is a keyword-based search engine for concepts and objects on the Semantic Web [19]. SWSE [20] and Sig.Ma [3] allow users to locate RDF entities via keyword search [17]. Some of the mentioned studies have also addressed the problem of query disambiguation, considering user preferences or heuristics [17].

Google has rolled out the new instant feature (Google Instant) in September 2010 [8]. As the user starts to type the first few letters of her search term, Google Instant automatically shows snippets of results for the most popular search term (the first suggested completion) that begins with those letters. The snippets appear below the box with suggested completions. As the user keeps typing, the snippets are dynamically updated. The user does not need to press enter or click the search button.

Google Instant helps the user to get better search results faster. Most importantly, seeing results as the user types her input helps her formulate a better search term by providing her with instant feedback [20]. However, Google Instant only displays result snippets for the first suggested completion in the drop-down suggestion box, even if the user moves the mouse over other suggestions. There are cases in which the user may be interested in making a choice between two suggested completions below the first one, but she cannot get instant feedback about them by using Google Instant. Rather she has to “make a commitment” to one of the suggested completions by clicking on it, which defeats the purpose of the instant feature, which is to minimize the number of user actions necessary to obtain a satisfactory result.

## Role of Ontology in search for Web pages

Recently, there has been a growing interest in Web searches that are intended to locate information that exists in the backend data bases of Web services. Web sites in E-commerce domains such as airfares, automobiles, books, car rentals, hotels, jobs, movies and music records usually store huge amounts of information, which is of interest to many users, in their backend databases [5].

Ontologies could play an important role in assisting users in their search for Web pages. Domain ontologies can be constructed that support users in their Web search efforts and that increase the number of relevant Web pages that are returned. To achieve this goal the Web information is combined with an ontology [5].

# Previous Work

## Ontology-Supported Web Search

The Ontology-Supported Web Search (OSWS) System for “famous people” provides search suggestions based on the user input, every time she types a new character. As seen in Figure 3, after the user completes the search term “Martina,” the system finds all the famous people in the knowledge base with “Martina” in their names. Additional background information about these famous people is extracted from the knowledge base for generating suggested completions. In this example, the tennis players “Martina Hingis” and “Martina Navratilova” and the singer “Martina McBride” are found. From the information related to these three famous people the suggested completions in the dropdown box are generated and displayed to the user [6].

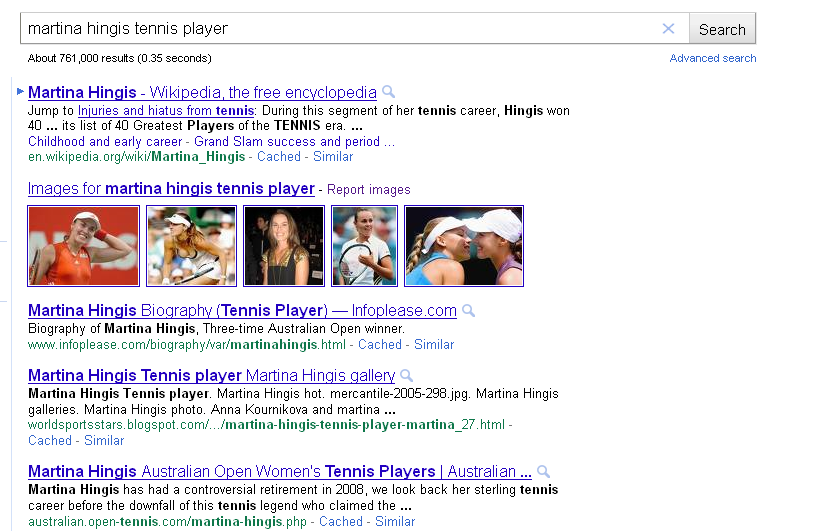


**Figure 3**: Interface of the OSWS system for search term "Martina" [6]

The processing is done as follows. For each concept of a famous person that is found in the ontology and has a name that matches the search term, all immediate neighbors along with their connecting relationships are retrieved. The current ontology in the OSWS system contains information about over 5000 musicians, more than 3000 basketball players and a selection of athletes from other kinds of sports. In the OSWS system, the first proposed suggestion about a famous person is always based on the class (modeling the occupation) of the person, which defines the name of the professional domain that the person belongs to.

Secondly, the system shows the result page snippets for one specific homonym every time the user hovers with the mouse on top of a suggested completion for a selectable time period, currently set to two seconds or longer. This helps the user to acquire a deeper and clearer understanding of the suggested term without having to make a choice. Google, at this point in time, does not automatically change the displayed snippets when the mouse is moved down to a lower suggested completion.

After user selects one suggestion and clicks the “Google Search” button, the system will redirect user to Google search result. In this example, when the user types “Martina”, OSWS system will show “Martina Hingis”, “Martina Navratilova” and “Martina McBride”, based on our ontology database. After user select “Martina Hingis”, OSWS will forward user’s selected query to Google search, as seen in Figure 4.



**Figure 4**: Google search result for "martina hingis tennis player"

# My Work

My responsibilities included coding, testing and designing the User Interface of the current OSWS system, which require the use of the JAVA programming language, GWT (Google Web Toolkit) and The Jena Ontology API. In detail, I have worked on the following tasks.

* Update system from GWT 1.6 to GWT 2.1
* Enhancement of User Interface
* Instant Search for Ontology-Supported Web Search
* Parallel Result Display for Homonyms
* Paging Support for Suggestions

## Update System from GWT 1.6 to GWT 2.1

The old OSWS system was using GWT 1.6, which was old and incompatible with the latest Google App Engine.

I removed the OSWS project dependency on Protégé OWL, which was incompatible with GWT 2.1. The current system does not use Protégé OWL anymore. It only depends on Jena 2.6.3, which can be downloaded from <http://sourceforge.net/projects/jena/files/Jena/Jena-2.6.3/>. Since there is an incompatibility issue between Jena 2.6.3 and GWT 2.1, I modified and recompiled the source code of Jena 2.6.3. Figure 5 shows the modifications I made. More details about how to recompile Jena 2.6.3 will be described in Appendix A: User Manual.

|  |  |
| --- | --- |
| **File** | **com/hp/hpl/jena/rdf/model/AnonId.java** |
| Old | **public** AnonId() {  **if** (JenaParameters.disableBNodeUIDGeneration) {  **synchronized** (AnonId.**class**) {  id = "A" + idCount++; // + rand.nextLong();  }  } **else** **if** (!UIDok) {  id = java.util.UUID.randomUUID().toString();  } **else** {  id = (**new** UID()).toString();  }  } |
| New | **public** AnonId() {  **if** (JenaParameters.*disableBNodeUIDGeneration*) {  **synchronized** (AnonId.**class**) {  id = "A" + *idCount*++; // + rand.nextLong();  }  } **else** {  id = java.util.UUID.*randomUUID*().toString();  }  } |
| **File** | **edu/njit/osws/server/OntologySuggestServiceImpl.java** |
| Old | String musicFile = getServletContext().getRealPath("/WEB-INF/Musician.pprj");  String basketballFile = getServletContext().getRealPath("/WEB-INF/Athlete.pprj");  ArrayList<String> errors = **new** ArrayList<String>();  Project project = **new** Project(musicFile, errors);  KnowledgeBase kb = project.getKnowledgeBase();  Cls cls = kb.getCls("Singer"); |
| New | URL musicFile = getServletContext().getResource("/WEB-INF/Musician.owl");  URL basketballFile = getServletContext().getResource("/WEB-INF/Athlete.owl");  OntModel onto = ModelFactory.*createOntologyModel*(OntModelSpec.*OWL\_MEM*, **null**);  onto.read(musicFile.toString(),  "RDF/XML-ABBREV");  String ns = onto.getNsPrefixURI("");  OntClass opus = (OntClass) onto.getOntClass(ns + "Singer"); |

**Figure 5**: Changes made for updating GWT

## Enhancement of User Interface

### System Layout

The layout of the old system was not well structured. I redesigned the whole layout of the system as Figure 6 below.



**Figure 6**: Layout of OSWS system

### Main Page Appearance

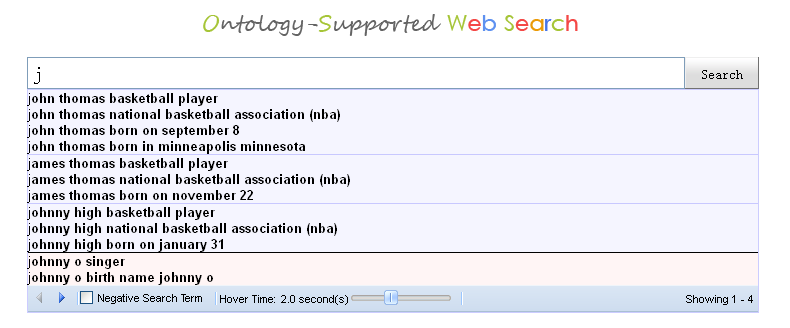
To make the whole system pretty and concise, I changed the system logo, removed unnecessary buttons and added a footer (copyright) section at the bottom of the page. Figure 7 shows the current main page of the OSWS system.



**Figure 7**: Current main page of OSWS system

### Search Result Page Appearance

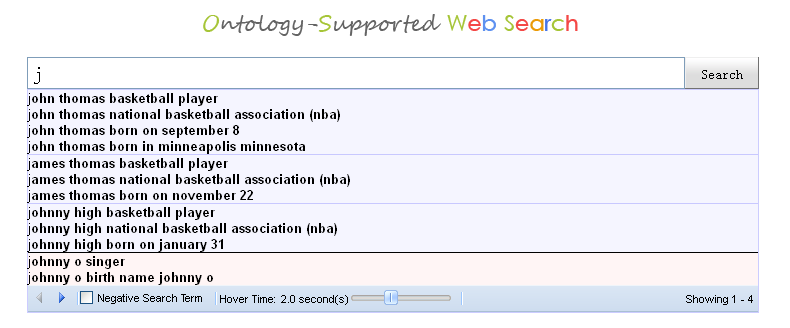
I also changed the search result page appearance, which can be seen in Figure 8. I used a smaller logo to save space for displaying results. Also, similar to Google search, I moved the search button to the right side of search input box.



**Figure 8**: The Current search page of the OSWS system

### Hover-Time Slider

As shown in Figure 9, I added a slider by using Ext GWT (see Appendix A), a Java UI component library for GWT. With this slider, the user can easily adjust the hover-time which refers to the time a user has to hover with the mouse on a suggestion to trigger a suggestion refresh. The default value is two seconds.



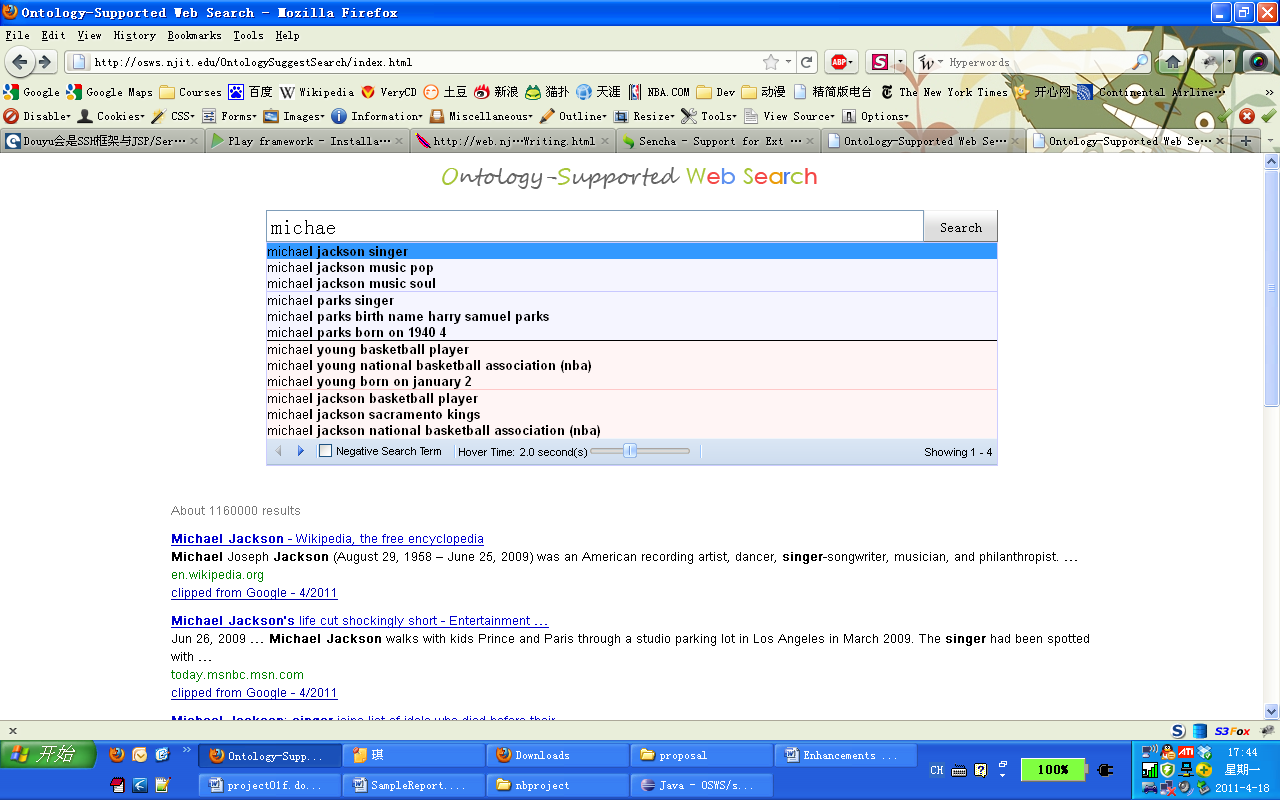
**Figure 9**: Slider for Hover-Time

## Instant Search for Ontology-Supported Web Search

The previous version of the Ontology-Supported Web Search system connected to the ontology database, parsed user queries and displayed relevant results in a categorized manner. I implemented an “Instant Search” [8] feature for a better user experience. The new interface will let users see the search result for the first suggested completion in real time. By predicting search queries and showing results before the user finishes typing, “Instant Search” can save 2-5 seconds per search [8]. This feature is similar to Google’s Instant Search shown in Figure 4. However, when moving the mouse down over the suggestions, the instant searches are not updated by Google. I implemented the suggest continuations in a way that whenever the mouse is hovering above a suggested continuation for a certain period of time, the Instant Search result display will be updated.

Figure 10 shows the Instant Search feature of the OSWS system. To achieve this feature, I used the Google AJAX Search 1.1 API to perform the search inside our program. Thus, in the current system, we can totally control the search results and customize the appearance of these results, which can be used in the implementation of the Instant Search feature. Figure 11 is the code snippet of performing a search by using the Google AJAX API.

To avoid an overload of the server and to better react to the user’s interests, we have defined a criterion for the minimum time required to be spent by the user hovering over a suggested completion. Thus, only if the user stays for a while on one suggested completion (without clicking), the corresponding results are displayed. The default hover time is set to two seconds, but as mentioned in section 3.2.3, this is a user-adjustable parameter.



**Figure 10**: OSWS Instant Search

|  |
| --- |
| SearchResultsHandler defaultHandler = **new** SearchResultsHandler() {  **public** **void** onSearchResults(**final** SearchResultsEvent event) {  JsArray<? **extends** Result> results = event.getResults();  clearSearchResult();  VerticalPanel resultColumn = **new** VerticalPanel();  resultColumn.setSpacing(10);  **long** resultCount = event.getSearch().getCursor().getEstimatedResultCount();  Label resultLabel = **new** Label("About "+resultCount+" results");  resultLabel.setStyleName("resultCount");  resultColumn.add(resultLabel);    **for** (**int** i = 0; i < results.length(); i++) {  WebResult result = (WebResult) results.get(i);  resultColumn.add(result.getHtml());  }  resultTable.add(resultColumn); }};  SearchControlOptions options = **new** SearchControlOptions();  WebSearch webSearch = **new** WebSearch();  options.add(webSearch);  searchControl = **new** SearchControl(options);  webSearch.setResultSetSize(ResultSetSize.*LARGE*);  searchControl.addSearchResultsHandler(defaultHandler); |

**Figure 11**: Performing a search by using the Google AJAX API

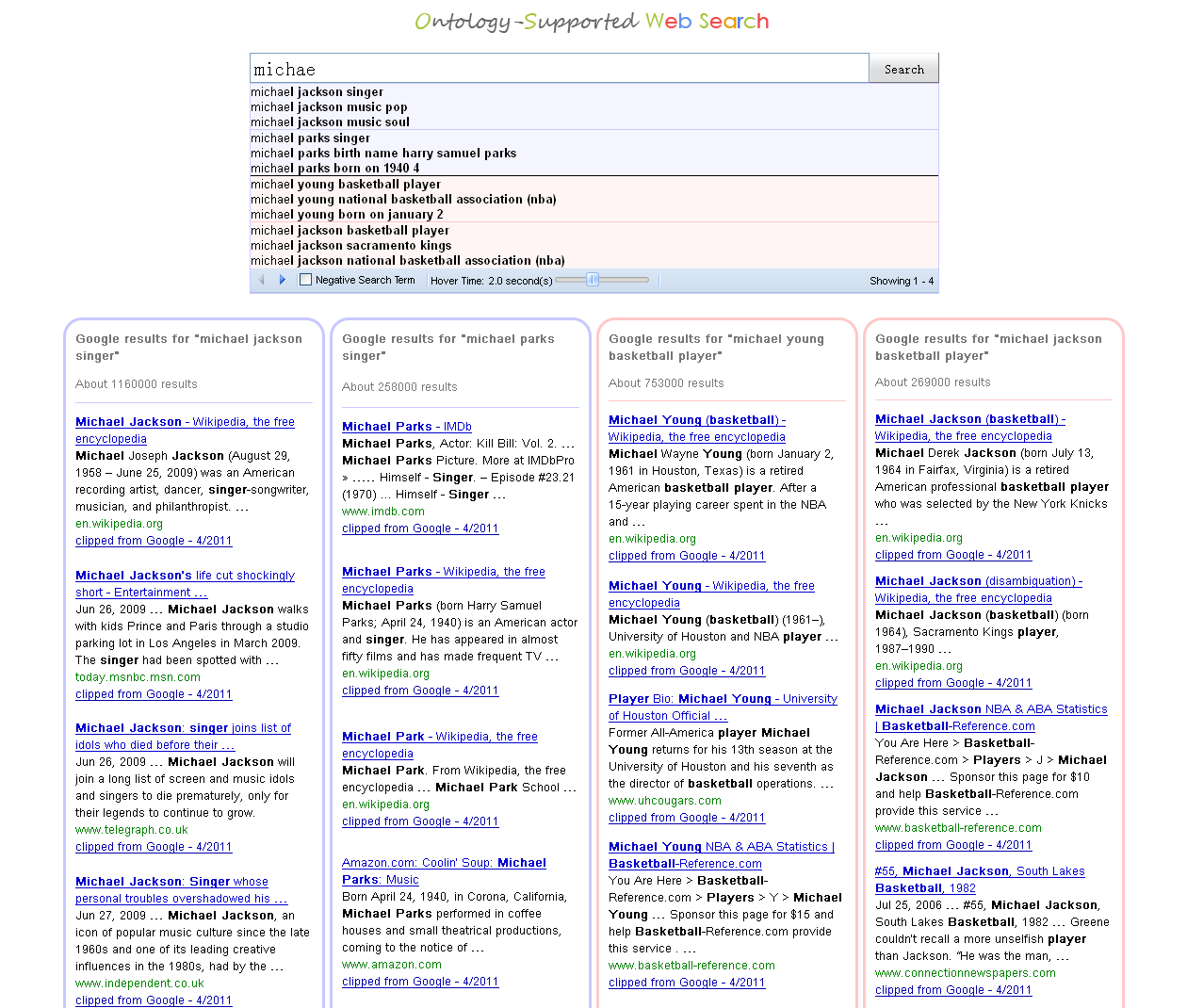
## Parallel Result Display for Homonyms

Previously, the Ontology-Supported Web Search engine could only show the results for one ontology term. Since we are getting categorized results from the ontology, it is better for the user to show different categorized search results in different frames.

As we categorized the different homonyms for a search term, such as singers, sports players, politicians, etc., we also divided the display screen below the suggested completions box into vertical panels with results for the different homonyms. Every vertical panel contains the Google search results for the first suggested completion of a different homonym.

As seen in Figure 12, after the user types “Michael,” four homonyms of famous “Michaels,” the singers Michael Jackson and Michael Parks and the basketball players Michael Young and Michael Jackson are found in our ontology and suggested to the user. Before the user decides which Michael fits his interest or moves his mouse to a lower suggested completion, the system instantly shows the results for the first suggestion of each of the four different Michaels. The display screen is tiled into four vertical panels, which are respectively the results for the search terms “Michael Jackson singer,” “Michael Parks singer,” “Michael Young basketball player” and “Michael Jackson basketball player.” Thus, the top down order in the suggested completions is reflected in the left-to-right order of the tiled windows containing result snippets.

The first suggested completion of each homonym is in most cases the occupation of the famous person. The returned snippets give the user richer and more detailed information about different homonyms to help her decide what result best fits her interests.



**Figure 12:** Parallel result display

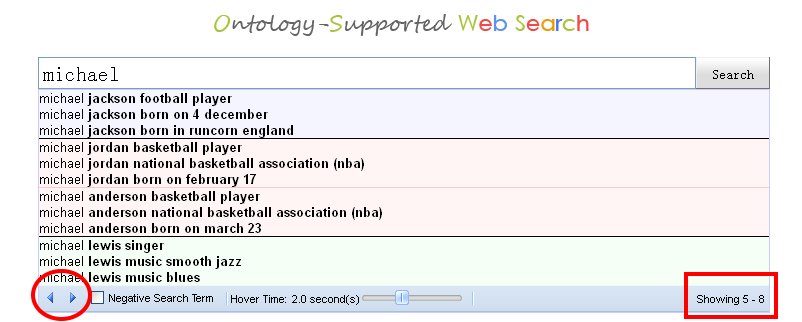
## Paging Support for Suggestions

Another feature I added is the paging for suggestions. In the old system, there were only 4 suggestions returned from the server side. I expanded the number to 20, and implemented a paging toolbar at the bottom of the suggestion drop-box. Figure 13 and 14 compare the suggestion drop-box between the old version and the current version.

When the user clicks on the left arrow in the red circle, then the display marked by the red rectangle changes to 1-4, and the drop-box displays the first four homonyms. Meanwhile, the left arrow changes its color to gray, since it has already reached the first homonym. If the user clicks on the right arrow, the display marked in the rectangle area changes to 9-12. The drop-box displays the third group of four homonyms.



**Figure 13:** Old Ontology-based suggested completions



**Figure 14:** New Ontology-based suggested completions

# Conclusions and Future Work

In order to improve the user experience, I have developed a new User Interface for the OSWS system. Following are the tasks I did for the new OSWS system:

1. Update system from GWT 1.6 to GWT 2.1: Modified the source code of Jena to make it work with GWT 2.1.
2. Enhancement of User Interface: Redesigned the layout of OSWS system, changed the system appearance.
3. Instant Search for Ontology-Supported Web Search: Used Google AJAX API to implement an “Instant Search” feature which is similar with Google’s Instant Search [8].
4. Parallel Result Display for Homonyms: Designed a feature that shows multiple search results for different homonyms.
5. Paging Support for Suggestions: Expanded the number of suggestions retrieved from ontology server from 4 to 20.

There are more improvements that can be implemented in the future. The current system does not support “browser history”, which means the “back” button of the browser does not work for the OSWS system. It is possible to support the “browser history” by using the History class from GWT. The current ontology is relatively small. It would be great if we could add more famous people from more domains. Our current ontology is loaded entirely at the beginning of the program, which is not efficient. We should consider using SPARQL for ontology querying in the future. Also we may consider integrating the ontology into database management system such as Oracle.

# References

1. Geller, J.: Definition of Ontology, [http://Web.njit.edu/~geller/what\_is\_an\_ontology.html](http://web.njit.edu/~geller/what_is_an_ontology.html), accessed Oct, 2010.
2. Gruber, T.: Ontology Entry in the Encyclopedia of Database Systems, Springer-Verlag (2008)
3. An, Y., Chun, S., Huang, K., Geller, J.: Enriching Ontology for Deep Web Search. In: DEXA, vol. 5181, pp 73-80. Lecture Notes in Computer Science. Turin, Italy (2008)
4. <http://vvww.cybergeo.eu/docannexe/image/8322/img-1.jpg>, accessed Oct, 2010.
5. Asher, J, Creating a Database from Deep Web Sources, NJIT, CS Department, MS Proposal, accessed Oct, 2010
6. Tian, T., Geller, J., Chun, S.A.: Improving Web Search Results for Homonyms by Suggesting Completions from an Ontology. In: 2nd International Workshop on Semantic Web Information Management (SWIM). Lecture Notes in Computer Science, pp. 175-186. Vienna, Austria (2010)
7. <http://en.wikipedia.org/wiki/Ontology_%28information_science%29>, accessed Oct, 2010
8. <http://www.google.com/instant/>, accessed Oct, 2010
9. Google Query Suggestion, <http://www.google.com/support/websearch/bin/answer.py?hl=en&answer=106230>, accessed Apr, 2011
10. Radev, D.R., Fan, W., Zhang, Z.: WebInEssence: A Personalized Web-Based Multi-Document Summarization and Recommendation System. In: NAACL Workshop on Automatic Summarization. Pittsburgh, PA (2001)
11. Al-Masri, E., Mahmoud, Q. H.: Discovering Web Services in Search Engines. In: IEEE Internet Computing, issue 3, pp. 74-77 (2008)
12. Lawrence, S.R.: Personalization of Web Search Results Using Term, Category, and Link-Based User Profiles. In: United States Patent Appl. 20100228715. Kind Code: A1. (2010)
13. Tian, T., Geller, J., Chun, S.A.: Predicting Web Search Hit Counts. In: 2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, pp.162-166. Toronto, Canada (2010)
14. Radlinski, F., Szummer, M., Craswell, N.: Inferring Query Intent from Reformulations and Clicks. In: WWW 2010. Raleigh, North Carolina, USA (2010)
15. Henzinger, M.: Search Technologies for the Internet. In: Science, vol. 317, no. 5837, pp. 468-471 (2007)
16. Fazzinga, B., Lukasiewicz, T.: Semantic Search on the Web. In: Semantic Web –Interoperability, Usability, Applicability, vol. 1, pp. 1-7 (2010)
17. Guha, R.V., McCool, R., Miller, E.: Semantic Search. In: WWW 2003, pp. 700-709. Budapest, Hungary (2003)
18. Cheng, G., Ge, W., Qu, Y.: Falcons: Searching and Browsing Entities on the Semantic Web. In: WWW 2008, pp. 1101-1102. Beijing China (2008)
19. Harth, A., Hogan, A., Delbru, R., Umbrich, J., O’Riain, S., Decker, S.: SWSE: Answer before Links!. In: Proceedings of Semantic Web Challenge (2007), CEUR Workshop (2007)
20. Tummarello, G., Cyganiak, R., Ctasta, M., Danielczyk, S., Delbru, R., Decker, S.: Sig.Ma: Live Views on the web of Data. In: WWW 2010, pp. 1301-1304. Raleigh, NC, USA (2010)
21. An, Y., Geller, J., Wu, Y., Chun, S: Semantic Deep Web: Automatic Attribute Extraction from the Deep Web Data Sources. In: Proceedings of the 2007 ACM Symposium on Applied computing, pp1667-1672. ACM-SAC, Seoul, Korea (2007)
22. An, Y., Chun, S., Huang, K., Geller, J.: Assessment for Ontology-Supported Deep Web Search. In: 2008 10th IEEE Conference on E-Commerce Technology and the Fifth IEEE Conference on Enterprise Computing, E-Commerce and E-Services, pp382-388. IEEE Computer Society (2008)
23. Ke, Y., Deng, L., Ng, W., Lee, D.: Web Dynamics and their Ramifications for the Development of Web Search Engines. In: Computer Networks: The International Journal of Computer and Telecommunications Networking – Web Dynamics, vol. 50, issue 10, pp. 1430-1447 (2006)

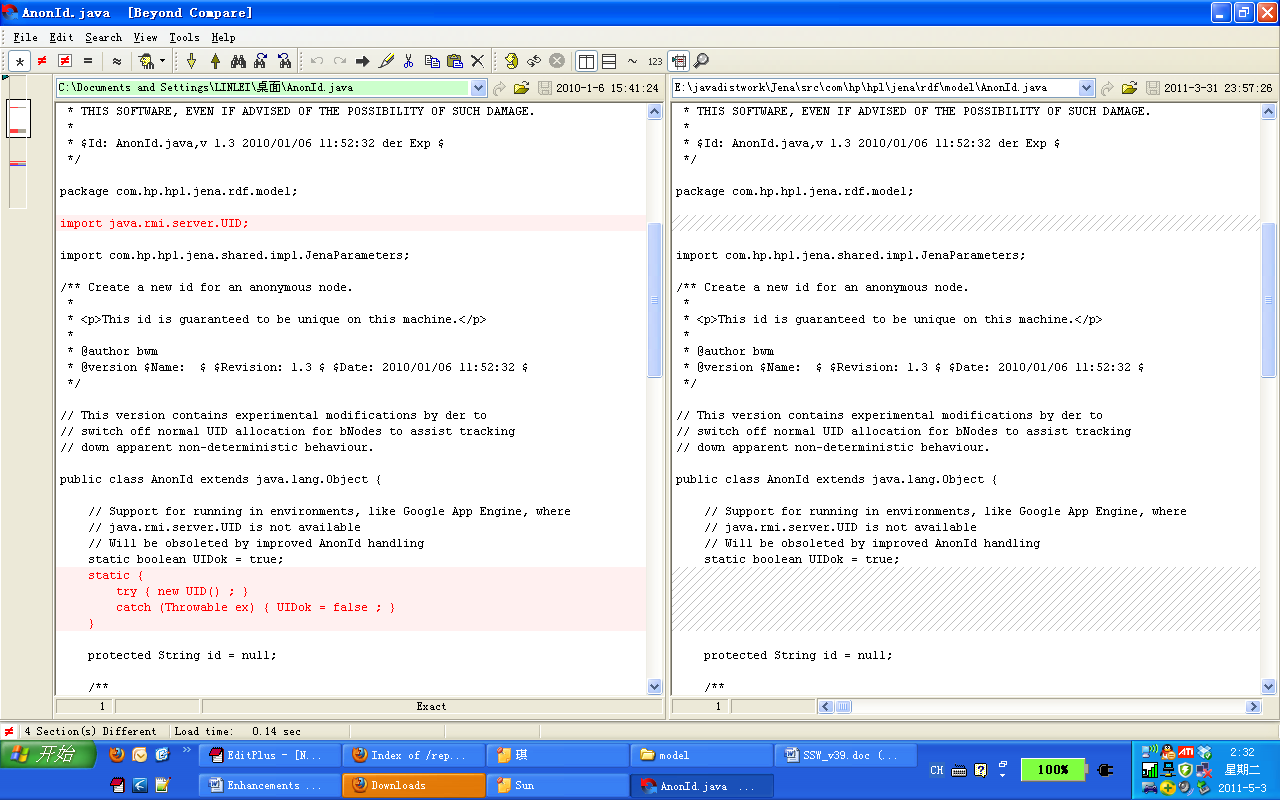
# Appendix A: User Manual

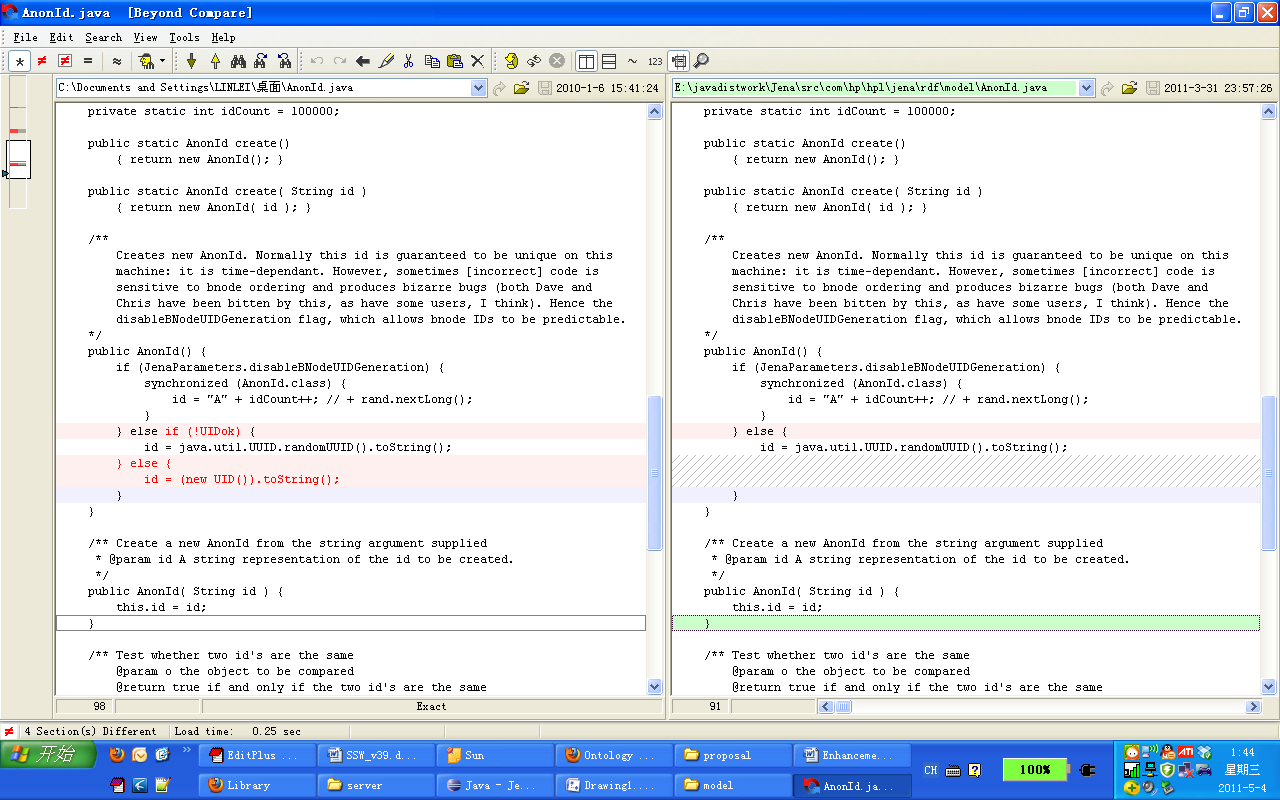
## Project Requirements

1. Eclipse 3.4 or higher
2. Google Plug-in for Eclipse
3. JDK 1.5
4. GWT 2.1
5. Ext GWT 2.2.1
6. Jena 2.6.3 (need to manually recompile, see Appendix A.2)

## Recompile Jena 2.6.3

1. Download Jena 2.6.3 from <http://sourceforge.net/projects/jena/files/Jena/Jena-2.6.3/> to a folder, for example: e:\jena\
2. Download Jena 2.6.3 source from <http://www.openjena.org/repo/com/hp/hpl/jena/jena/2.6.3/jena-2.6.3-sources.jar>, unzip the source package to a folder, such as e:\jena-src\
3. Go to directory: e:\jena-src\com\hp\hpl\jena\rdf\model\. We can find the AnonId.java file.
4. In AnonId.java, remove code marked in red color:





1. Use JDK 1.5 to recompile the modified AnonId.java to AnonId.class by using following commands.

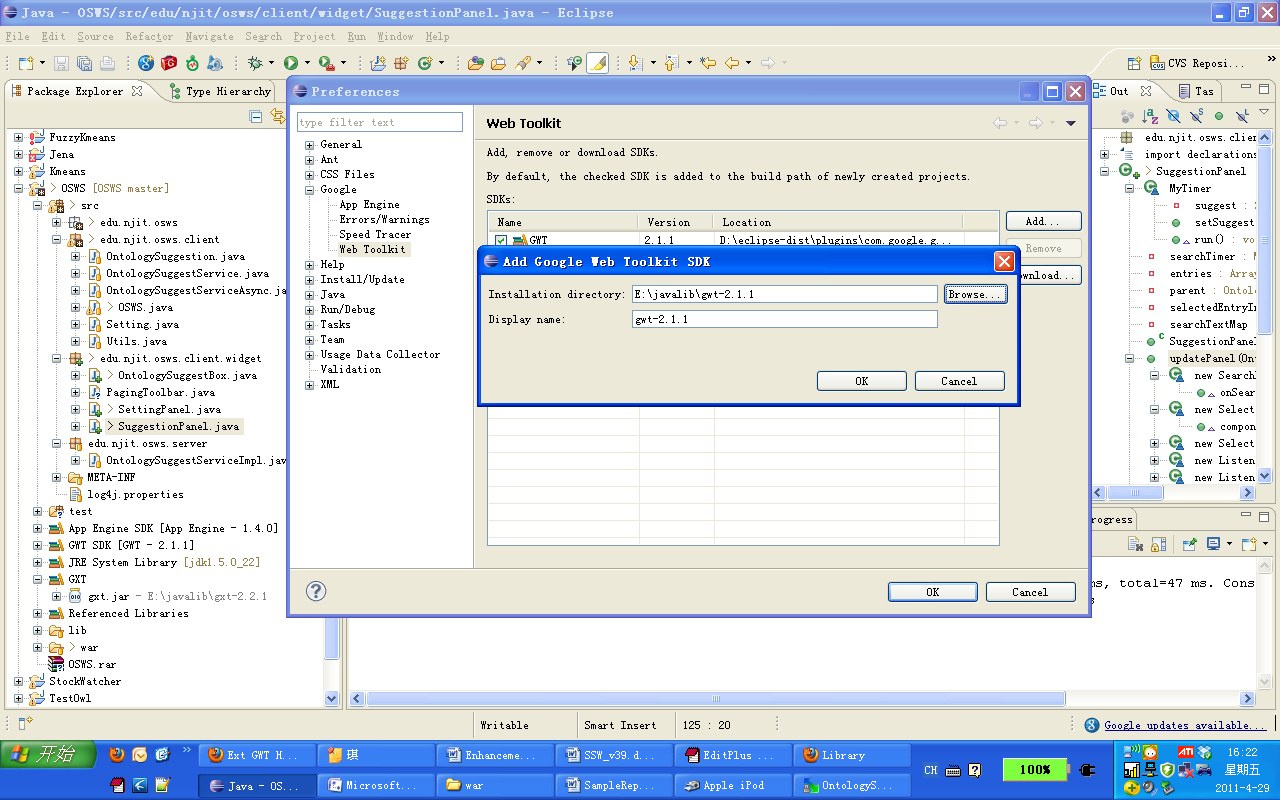
|  |
| --- |
| cd e:\jena-src\ |
| "c:\Program Files\Java\jdk1.5.0\_22\bin\javac.exe" com/hp/hpl/jena/rdf/model/AnonId.java |

The compiled AnonId.class can be found from e:\jena-src\com\hp\hpl\jena\rdf\.

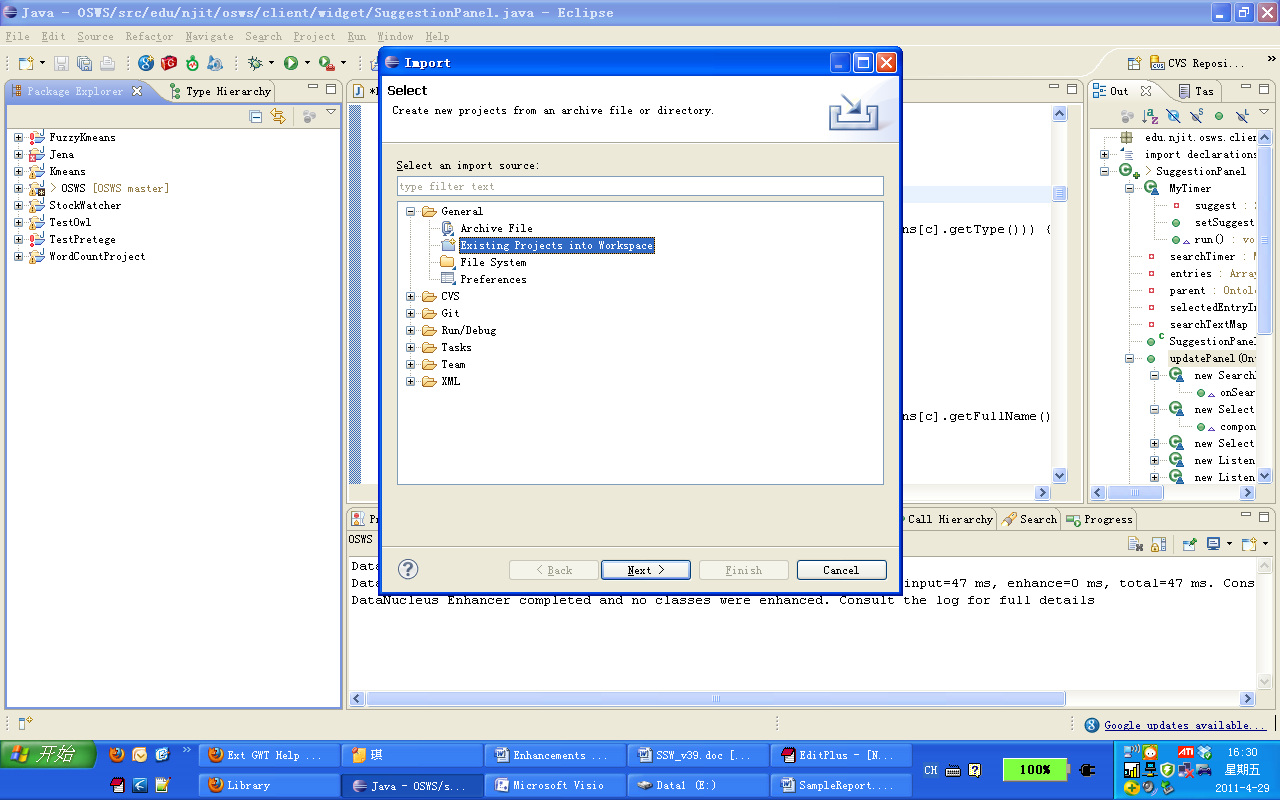
1. Replace the AnonId.class in downloaded jena-2.6.3.jar with the modified one we just compiled. One simple approach to do that is using WinRAR to open jena-2.6.3.jar and replace the AnonId.class.

## Setup Project in Eclipse

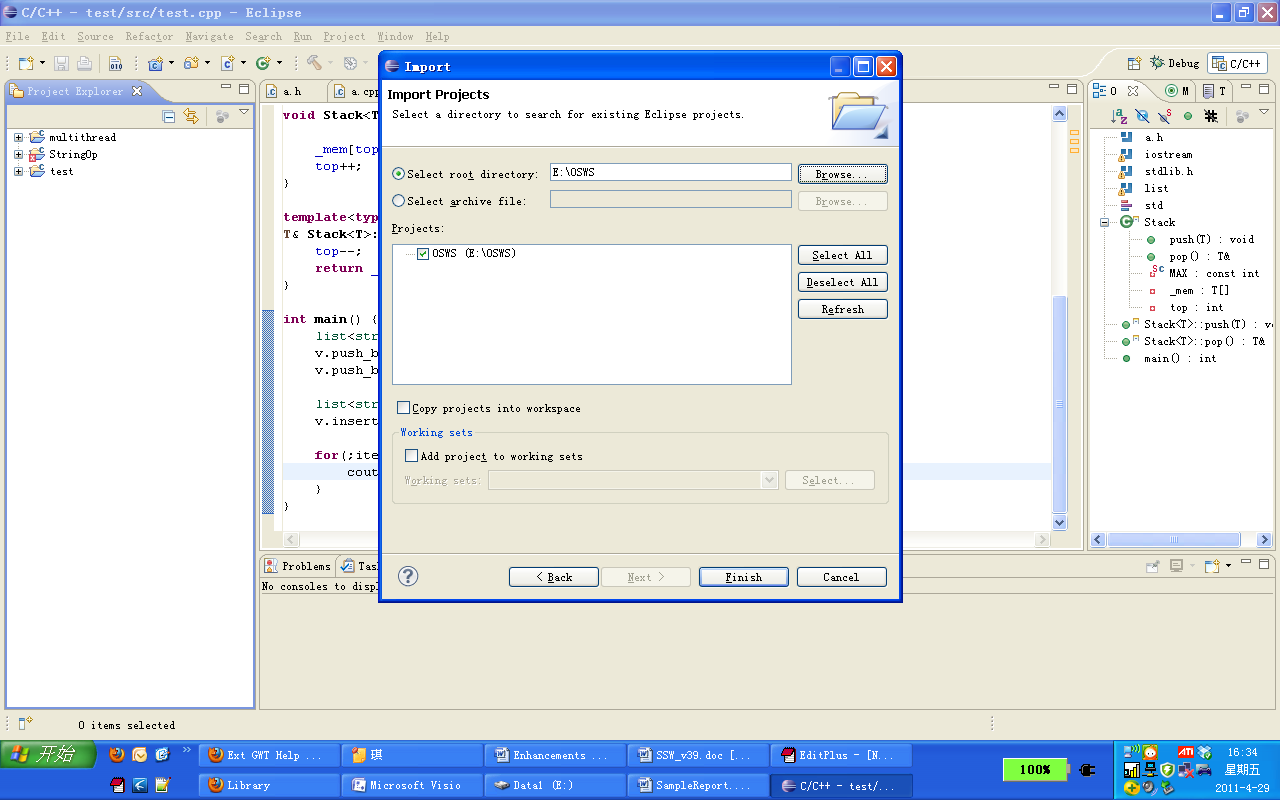
1. Install Google Plug-in for Eclipse, there is an install guide located at <http://code.google.com/webtoolkit/usingeclipse.html>
2. Download GWT SDK 2.1 from <http://google-web-toolkit.googlecode.com/files/gwt-2.1.1.zip>. Unzip the package to a folder such as e:\javalib\gwt-2.1.1\
3. Open Eclipse. Go to Windows -> Preferences -> Google -> Web Kit. Click on “Add…” to add the downloaded GWT SDK 2.1 into eclipse. And check the checkbox in front of GWT 2.1 to switch the default GWT to GWT 2.1.



1. Download Ext GWT 2.2.1 from <http://www.sencha.com/products/extgwt/>. Follow the guide <http://www.sencha.com/helpcenter/index.jsp?topic=/com.extjs.gxt.help/html/gettingstarted/setup.html> to setup it in Eclipse.
2. Unpack the source package to a folder, for example, e:\osws
3. In Eclipse, Select File->Import->General->Existing Project into Workspace. Click “next”



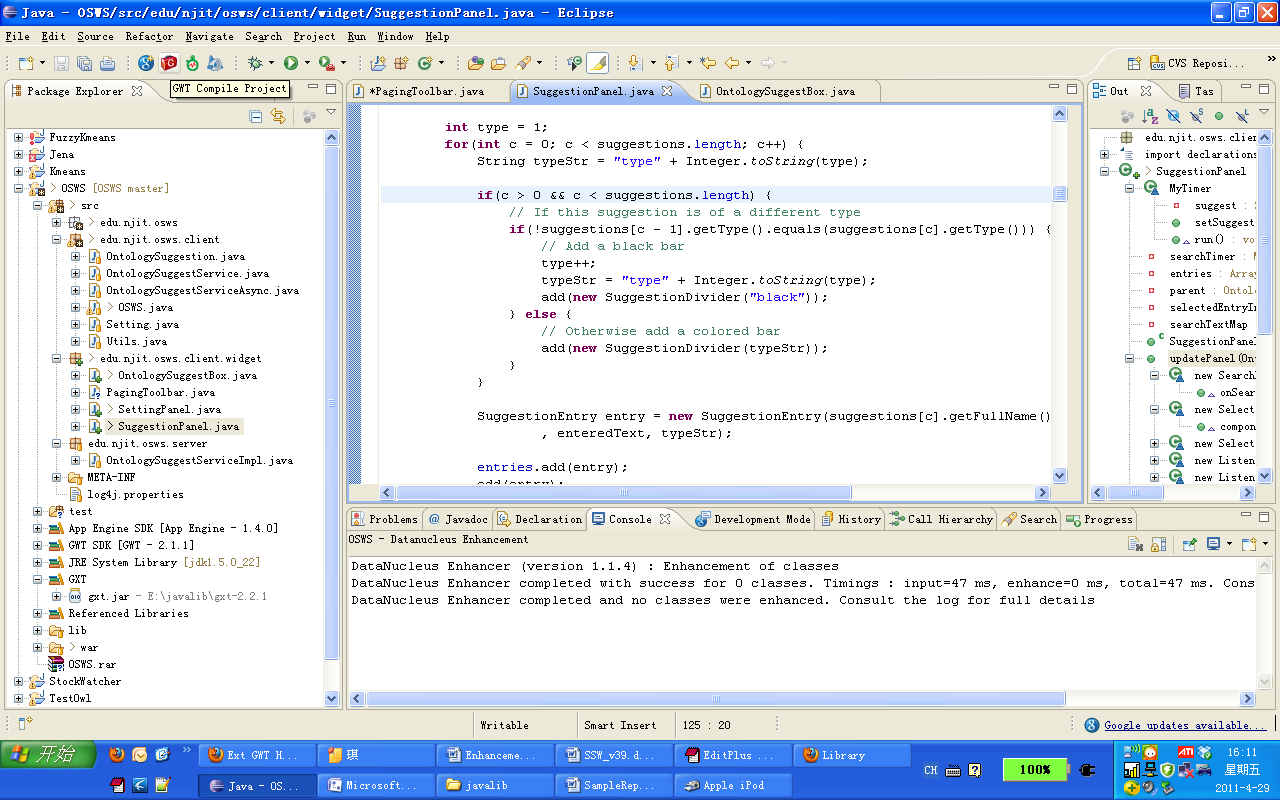
Click “Browse…” to locate the unzipped source folder, and click “finish”.



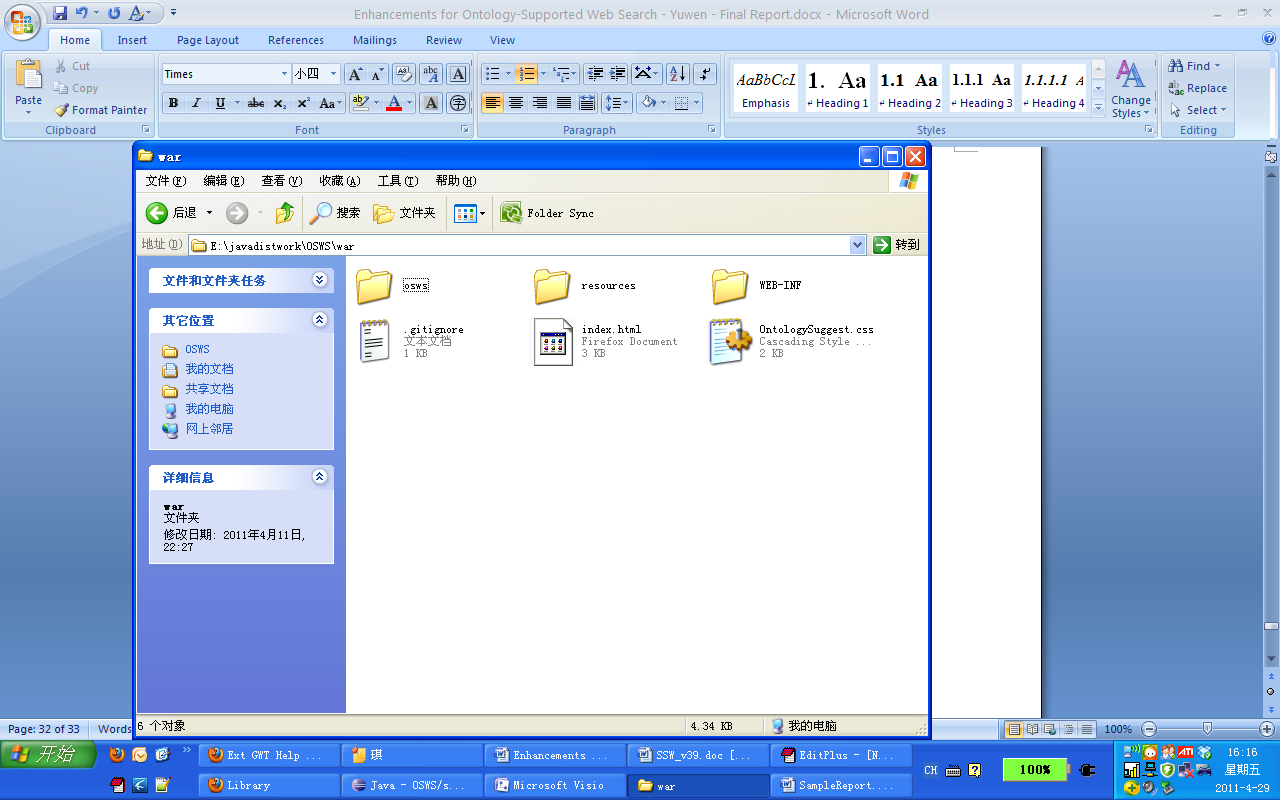
1. Switch the project JDK to 1.5 in “Build Path” panel.
2. The OSWS project is successfully imported.

## Compiling and Deploying project to AFS

1. Click on “GWT Compile Project” icon to compile the OSWS project



1. Copy all the contents in the “war” folder to AFS: /afs/cad.njit.edu/research/ccs/geller/osws/webapps/OntologySuggestSearch.



1. Go to <http://osws.njit.edu> to verify the deployment.

# Appendix B: Source Code

## index.html

|  |
| --- |
| <!doctype html>  <!-- The DOCTYPE declaration above will set the -->  <!-- browser's rendering engine into -->  <!-- "Standards Mode". Replacing this declaration -->  <!-- with a "Quirks Mode" doctype may lead to some -->  <!-- differences in layout. -->  <html>  <head>  <meta http-equiv="content-type" content="text/html; charset=UTF-8">  <!-- -->  <!-- Consider inlining CSS to reduce the number of requested files -->  <!-- -->  <link type="text/css" rel="stylesheet" href="OntologySuggest.css">  <link rel="stylesheet" type="text/css" href="resources/css/gxt-all.css" />  <script language='javascript' src='resources/flash/swfobject.js'></script>  <!-- -->  <!-- Any title is fine -->  <!-- -->  <title>Ontology-Supported Web Search</title>    <!-- -->  <!-- This script loads your compiled module. -->  <!-- If you add any GWT meta tags, they must -->  <!-- be added before this line. -->  <!-- -->  <script type="text/javascript" language="javascript" src="osws/osws.nocache.js"></script>  </head>  <!-- -->  <!-- The body can have arbitrary html, or -->  <!-- you can leave the body empty if you want -->  <!-- to create a completely dynamic UI. -->  <!-- -->  <body>  <!-- OPTIONAL: include this if you want history support -->  <iframe src="javascript:''" id="\_\_gwt\_historyFrame" tabIndex='-1' style="position:absolute;width:0;height:0;border:0"></iframe>    <!-- RECOMMENDED if your web app will not function without JavaScript enabled -->  <noscript>  <div style="width: 22em; position: absolute; left: 50%; margin-left: -11em; color: red; background-color: white; border: 1px solid red; padding: 4px; font-family: sans-serif">  Your web browser must have JavaScript enabled  in order for this application to display correctly.  </div>  </noscript>    </body>  </html> |

## OntologySuggest.css

|  |
| --- |
| root {  display: block;  }  html {  overflow: -moz-scrollbars-vertical;  }  .searchButton {  padding-left: 100px;  font-size: 16pt;  }  .resultTable {  margin-top: 20px;  }  .query {  color: #777777;  font-weight: bold;  }  .resultCount {  color: #777777;  font-weight: italic;  }  .suggestStyle {  border-right-style: solid;  border-left-style: solid;  border-bottom-style: solid;  border-top-style: solid;  border-right-width: 1px;  border-left-width: 1px;  border-bottom-width: 1px;  border-top-width: 1px;  border-right-color: #ccccff;  border-left-color: #ccccff;  border-bottom-color: #ccccff;  border-top-color: #ccccff;  }  .suggestionPanelStyle {    }  .suggestionTextStyle {  font-size: small;  font-family: Arial,Helvetica,sans-serif;  text-align: left;  }  .type1-normal {  background-color: #f5f5ff;  }  .type2-normal {  background-color: #fff5f5;  }  .type3-normal {  background-color: #f5fff5;  }  .type4-normal {  background-color: #fffff5;  }  .highlight {  background-color: #3399ff;  }  .divider-black {  background-color: #000000;  }  .divider-type1 {  background-color: #c8c8ff;  }  .divider-type2 {  background-color: #ffc8c8;  }  .divider-type3 {  background-color: #c8ffc8;  }  .divider-type4 {  background-color: #ffffc8;  }  .textboxStyle {  font-size: 16pt;  }  .settingPanel {  margin-top: 5px;  margin-bottom: 5px;  }  .footer {  font-size: small;  margin-top:30px;  margin-bottom:5px;  font-family: Arial,Helvetica,sans-serif;  color: #888888;  }  .border-1 {  border-style: solid;  border-width:1;  border-color: #c8c8ff;  }  .border-2 {  border-style: solid;  border-width:1;  border-color: #ffc8c8;  }  .border-3 {  border-style: solid;  border-width:1;  border-color: #c8ffc8;  }  .border-4 {  border-style: solid;  border-width:1;  border-color: #ffffc8;  }  .rounded-corners {  -moz-border-radius: 20px;  -webkit-border-radius: 20px;  -khtml-border-radius: 20px;  border-radius: 20px;  } |

## OSWS.gwt.xml

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <module rename-to='osws'>  <!-- Inherit the core Web Toolkit stuff. -->  <inherits name='com.google.gwt.user.User' />  <inherits name='com.extjs.gxt.ui.GXT'/>    <!-- Inherit the default GWT style sheet. You can change -->  <!-- the theme of your GWT application by uncommenting -->  <!-- any one of the following lines. -->  <inherits name='com.google.gwt.user.theme.standard.Standard' />  <!-- <inherits name='com.google.gwt.user.theme.chrome.Chrome'/> -->  <!-- <inherits name='com.google.gwt.user.theme.dark.Dark'/> -->  <!-- Other module inherits -->  <inherits name='com.google.gwt.search.Search' />  <!-- Specify the app entry point class. -->  <entry-point class='edu.njit.osws.client.OSWS' />  <!-- Specify the paths for translatable code -->  <source path='client' />  <source path='client/widget' />  <source path='shared' />  <script src="http://www.google.com/uds/api?file=uds.js&amp;v=1.0&amp;gwt=1"/>  </module> |

## OntologySuggestService.java

|  |
| --- |
| package edu.njit.osws.client;  import com.google.gwt.user.client.rpc.RemoteService;  import com.google.gwt.user.client.rpc.RemoteServiceRelativePath;  @RemoteServiceRelativePath("ontologysuggestservice")  public interface OntologySuggestService extends RemoteService {  public OntologySuggestion [] getSuggestions(String s, boolean negativeTerms, int offset);  } |

## OntologySuggestServiceAsync.java

|  |
| --- |
| package edu.njit.osws.client;  import com.google.gwt.user.client.rpc.AsyncCallback;  public interface OntologySuggestServiceAsync {  public void getSuggestions(String s, boolean negativeTerms,int offset,  AsyncCallback<OntologySuggestion[]> callback);  } |

## OntologySuggestServiceImpl.java

|  |
| --- |
| package edu.njit.osws.server;  import java.io.PrintWriter;  import java.io.StringWriter;  import java.io.Writer;  import java.net.URL;  import java.util.ArrayList;  import java.util.Collections;  import java.util.Comparator;  import java.util.HashMap;  import java.util.Iterator;  import com.google.gwt.core.client.GWT;  import com.google.gwt.user.server.rpc.RemoteServiceServlet;  import com.hp.hpl.jena.ontology.OntClass;  import com.hp.hpl.jena.ontology.OntModel;  import com.hp.hpl.jena.ontology.OntModelSpec;  import com.hp.hpl.jena.ontology.OntResource;  import com.hp.hpl.jena.rdf.model.ModelFactory;  import com.hp.hpl.jena.rdf.model.Statement;  import com.hp.hpl.jena.rdf.model.StmtIterator;  import edu.njit.osws.client.OntologySuggestService;  import edu.njit.osws.client.OntologySuggestion;  @SuppressWarnings("serial")  public class OntologySuggestServiceImpl extends RemoteServiceServlet implements OntologySuggestService {  private static ArrayList<OntologySuggestion> suggestions;  private static TypeHierarchy typeHierarchy;  private static boolean loading = false;  private static HashMap<String, Integer> relPriorityMap;  private class TypeHierarchy {  private String superclass;  private ArrayList<String> subclass = new ArrayList<String>();  public void setSuperClass(String s) {  superclass = s;  }  public void addSubClass(String s) {  subclass.add(s);  }  public String getSuperClass() {  return superclass;  }  public boolean hasSubClass(String s) {  if (subclass.contains(s)) {  return true;  }  return false;  }  }  private String superClass(String sub) {  if (typeHierarchy.hasSubClass(sub)) {  return typeHierarchy.getSuperClass();  }  return "";  }        // public PagingLoadResult<OntologySuggestion> getSuggestions(String searchTerm, boolean negativeTerms,PagingLoadConfig config) {  // OntologySuggestion[] su = getSuggestions(searchTerm, negativeTerms);  // int count = su.length;  // return new BasePagingLoadResult<OntologySuggestion>(Arrays.asList(su), config.getOffset(), (int)count);  // }  public OntologySuggestion[] getSuggestions(String searchTerm, boolean negativeTerms, int offset) {  // Load the data. Needs to be improved.  if (suggestions == null && !loading) {  loading = true;  System.out.println("LOADING SUGGESTIONS");  relPriorityMap = new HashMap<String, Integer>();  relPriorityMap.put("plays", 1);  relPriorityMap.put("inTeam", 1);  relPriorityMap.put("hasBirthName", 2);  relPriorityMap.put("inLeague", 2);  relPriorityMap.put("hasBirthDate", 3);  relPriorityMap.put("hasBirthPlace", 4);  suggestions = loadOntologySuggestions();  loading = false;  } else if (suggestions == null && loading == true) {  return new OntologySuggestion[0];  }  String line = searchTerm.toLowerCase();  ArrayList<OntologySuggestion> results = new ArrayList<OntologySuggestion>();  // For every concept in our ontology  for (OntologySuggestion a : suggestions) {  String[] splitLine = line.split(" ");  String[] nameSplit = a.getFullName().split(" ");  // If the search has name and type  if (line.contains(a.getFullName()) && splitLine.length > nameSplit.length) {  String name = a.getFullName();  String type = line.substring(line.indexOf(name) + name.length()).trim();  if (a.getType().startsWith(type) || superClass(a.getType()).startsWith(type)) {  results.add(a);  }  } else {  if (a.getFullName().startsWith(line)) {  results.add(a);  }  }  }  ArrayList<OntologySuggestion> shortenedResults = new ArrayList<OntologySuggestion>();  // We only want the 4 results  for (int c = offset; c<results.size() && c < offset+4; c++) {  shortenedResults.add(results.get(c));  }  results = shortenedResults;  ArrayList<OntologySuggestion> finalResults = new ArrayList<OntologySuggestion>();  // Sort the results by their classes  while (results.size() > 0) {  OntologySuggestion suggestion = results.remove(0);  finalResults.add(suggestion);  for (int c = 0; c < results.size(); c++) {  if (results.get(c).getType().equals(suggestion.getType())) {  finalResults.add(results.remove(c));  }  }  }  results = finalResults;  // If we want negative search terms, get them  if (negativeTerms && results.size() > 1) {  results = getNegativeSearchResults(results);  }  GWT.log("---> "+results.size());  return results.toArray(new OntologySuggestion[results.size()]);  }  private ArrayList<OntologySuggestion> getNegativeSearchResults(  ArrayList<OntologySuggestion> suggestions) {  ArrayList<OntologySuggestion> negativeSuggestions = new ArrayList<OntologySuggestion>();  // While there are non-negative suggestions  for (OntologySuggestion suggestion : suggestions) {  ArrayList<OntologySuggestion> sameConceptNameSuggestions =  new ArrayList<OntologySuggestion>();  // Find all of the other suggestions with the same name as last removed  for (OntologySuggestion otherSuggestion : suggestions) {  if (suggestion != otherSuggestion &&  suggestion.getFullName().equalsIgnoreCase(otherSuggestion.getFullName())) {  sameConceptNameSuggestions.add(otherSuggestion);  }  }  String type = suggestion.getType();  boolean firstNegType = true;  // Loop through the other concepts that are not the current one and get their type  for (OntologySuggestion sug : sameConceptNameSuggestions) {  if (suggestion != sug && !suggestion.getType().equalsIgnoreCase(sug.getType())) {  type += " [" + (firstNegType ? "but" : "and") + " not] \"" + sug.getType() + "\"";  firstNegType = false;  }  }  OntologySuggestion negativeSuggestion = new OntologySuggestion();  negativeSuggestion.setFullName(suggestion.getFullName());  negativeSuggestion.setHitCount(suggestion.getHitCount());  negativeSuggestion.setType(type);  // For all of the attributes  for (String rel : suggestion.getAttributeList()) {  String newRel = rel;  String relType = rel.substring(0, rel.indexOf(" ")).trim().toLowerCase();  boolean firstNegAttr = true;  // Compare them against the attributes of the other suggestions of the  // same concept name  for (OntologySuggestion sug : sameConceptNameSuggestions) {  ArrayList<String> otherRel = sug.getAttributeList();  for (String attr : otherRel) {  String otherRelType = attr.substring(0, attr.indexOf(" ")).trim().toLowerCase();  if (otherRelType.equals(relType)) {  newRel += " [" + (firstNegAttr ? "but" : "and") + " not] \"" + attr.substring(attr.indexOf(" ")).trim() + "\"";  firstNegAttr = false;  }  }  }  negativeSuggestion.addAttribute(newRel);  }  negativeSuggestions.add(negativeSuggestion);  }  return negativeSuggestions;  }  private ArrayList<OntologySuggestion> loadOntologySuggestions() {  ArrayList<OntologySuggestion> data = new ArrayList<OntologySuggestion>();  try {  URL musicFile = getServletContext().getResource("/WEB-INF/Musician.owl");  URL basketballFile = getServletContext().getResource("/WEB-INF/Athlete.owl");  OntModel onto = ModelFactory.createOntologyModel(  OntModelSpec.OWL\_MEM, null);  onto.read(musicFile.toString(),  "RDF/XML-ABBREV");  String ns = onto.getNsPrefixURI("");  OntClass opus = (OntClass) onto.getOntClass(ns + "Singer");  for (Iterator<? extends OntResource> bs = opus.listInstances(); bs  .hasNext();) {  OntResource ins = (OntResource) bs.next();  data.add(createSuggestion(ins));  }  onto.reset();  onto.read(basketballFile.toString(),  "RDF/XML-ABBREV");  ns = onto.getNsPrefixURI("");  opus = (OntClass) onto.getOntClass(ns + "Athlete");  typeHierarchy = new TypeHierarchy();  typeHierarchy.setSuperClass("athlete");  for (Iterator<OntClass> it = opus.listSubClasses(); it.hasNext();) {  OntClass subclass = (OntClass) it.next();  String className = processName(subclass.getLocalName());  typeHierarchy.addSubClass(className);  for (Iterator<? extends OntResource> bs = subclass.listInstances(); bs  .hasNext();) {  OntResource ins = (OntResource) bs.next();  data.add(createSuggestion(ins));  }  }  Collections.sort(data, new Comparator<OntologySuggestion>() {  public int compare(OntologySuggestion a, OntologySuggestion b) {  if (a.getHitCount() > b.getHitCount()) {  return -1;  } else if (a.getHitCount() < b.getHitCount()) {  return 1;  } else {  return 0;  }  }  });  for (OntologySuggestion suggestion : data) {  ArrayList<String> attributes = suggestion.getAttributeList();  Collections.sort(attributes, new Comparator<String>() {  public int compare(String a, String b) {  if (relPriorityMap.get(a.substring(0, a.indexOf(" "))) < relPriorityMap  .get(b.substring(0, b.indexOf(" ")))) {  return -1;  } else {  return 1;  }  }  });  }  } catch (Exception e) {  // TODO Auto-generated catch block    OntologySuggestion fake = new OntologySuggestion();  fake.setFullName("error");  fake.setType(getStackTrace(e));  data.add(fake);  }  return data;  }    public static String getStackTrace(Throwable aThrowable) {  final Writer result = new StringWriter();  final PrintWriter printWriter = new PrintWriter(result);  aThrowable.printStackTrace(printWriter);  return result.toString();  }  private OntologySuggestion createSuggestion(OntResource ins) {  OntologySuggestion suggestion = new OntologySuggestion();  String name = ins.getLocalName();  if(name.startsWith("\_")){  name = ins.getNameSpace().replaceFirst(".\*\\.owl#(.\*)", "$1")+ins.getLocalName();  }  name = processName(name);  suggestion.setFullName(name);  for (StmtIterator it = ins.listProperties(); it.hasNext();) {  Statement res = (Statement) it.next();  String attrib = res.getPredicate().getLocalName();  if (attrib.equals("type"))  suggestion  .setType(processName(ins.getRDFType().getLocalName()));  else if (attrib.equals("hasHitCount"))  suggestion.setHitCount(Long.parseLong(res.getLiteral()  .getString()));  else {  if (!attrib.trim().equals("")  && !res.getLiteral().getString().trim().equals(""))  suggestion.addAttribute(attrib + " "  + processName(res.getLiteral().getString()));  }  }  return suggestion;  }  private String processName(String name) {  String[] temp = name.split("[\_]");  String r = temp[0];  int index = temp.length;  try {  Integer.parseInt(temp[index - 1]);  index--;  } catch (Exception e) {  //e.printStackTrace();  } finally {  for (int i = 1; i < index; i++) {  if (temp[i].length() > 0) {  r += " " + temp[i].trim();  }  }  }  return r.trim().toLowerCase();  }  } |

## OntologySuggestion.java

|  |
| --- |
| package edu.njit.osws.client;  import com.google.gwt.user.client.rpc.IsSerializable;  import java.util.ArrayList;  public class OntologySuggestion implements IsSerializable {  private String fullName;  private String type;  private ArrayList<String> attributes;  private int nameUniqueIdentifier;  private long hitCount;  public OntologySuggestion() {  this.fullName = "";  this.type = "";  this.attributes = new ArrayList<String>();  this.nameUniqueIdentifier = 0;  this.hitCount = 0;  }  public OntologySuggestion(String fullName, String type, ArrayList<String> attributes,  int nameIdentifier, long hitCount) {  this.fullName = fullName;  this.type = type;  this.attributes = attributes;  this.nameUniqueIdentifier = nameIdentifier;  this.hitCount = hitCount;  }  public String getFullName() {  return fullName;  }  public void setFullName(String name) {  this.fullName = name;  }  public String getType() {  return type;  }  public void setType(String type) {  this.type = type;  }  public ArrayList<String> getAttributeList() {  return attributes;  }  public void addAttribute(String attribute) {  attributes.add(attribute);  }  public int getNameIdentifier() {  return nameUniqueIdentifier;  }  public void setHitCount(long value) {  this.hitCount = value;  }    public long getHitCount() {  return hitCount;  }  } |

## OSWS.java

|  |
| --- |
| package edu.njit.osws.client;  import com.google.gwt.core.client.EntryPoint;  import com.google.gwt.core.client.GWT;  import com.google.gwt.user.client.ui.RootPanel;  import com.google.gwt.user.client.ui.VerticalPanel;  import edu.njit.osws.client.widget.OntologySuggestBox;  /\*\*  \* Entry point classes define <code>onModuleLoad()</code>.  \*/  public class OSWS implements EntryPoint {  /\*\*  \* The message displayed to the user when the server cannot be reached or  \* returns an error.  \*/  private static final String SERVER\_ERROR = "An error occurred while "  + "attempting to contact the server. Please check your network "  + "connection and try again.";  /\*\*  \* Create a remote service proxy to talk to the server-side Greeting  \* service.  \*/  private final OntologySuggestServiceAsync ontologySuggestService = GWT  .create(OntologySuggestService.class);      /\*\*  \* This is the entry point method.  \*/  public void onModuleLoad() {  // ((ServiceDefTarget)ontologySuggestService).setServiceEntryPoint("http://osws.njit.edu/OntologySuggestSearch/osws/ontologysuggestservice");  VerticalPanel content = new VerticalPanel();  content.setHorizontalAlignment(VerticalPanel.ALIGN\_CENTER);  content.setWidth("100%");    final OntologySuggestBox osb = new OntologySuggestBox();  content.add(osb);    RootPanel.get().add(content);    osb.giveFocus();  }  } |

## Setting.java

|  |
| --- |
| package edu.njit.osws.client;  public class Setting {  public static int HOVER\_TIME = 2000;  public static int UPDATE\_TIME = 750;  public static boolean NEGATIVE\_TERMS = false;  public static final int TERMS\_MAX = 20;  public static final int TERMS\_UNIT = 4;    public static final String VERSION = "1.3.1";  } |

## Utils.java

|  |
| --- |
| package edu.njit.osws.client;  public class Utils {  public static String term2Query(String terms){  if(Setting.NEGATIVE\_TERMS)  return terms.replaceAll("\\[but not\\] ", " -").replaceAll(  "\\[and not\\] ", " -");  else  return terms;  }  } |

## OntologySuggestBox.java

|  |
| --- |
| package edu.njit.osws.client.widget;  import com.google.gwt.core.client.GWT;  import com.google.gwt.core.client.JsArray;  import com.google.gwt.event.dom.client.ClickEvent;  import com.google.gwt.event.dom.client.ClickHandler;  import com.google.gwt.event.dom.client.KeyCodes;  import com.google.gwt.event.dom.client.KeyUpEvent;  import com.google.gwt.event.dom.client.KeyUpHandler;  import com.google.gwt.search.client.Result;  import com.google.gwt.search.client.ResultSetSize;  import com.google.gwt.search.client.Search.Page;  import com.google.gwt.search.client.SearchControl;  import com.google.gwt.search.client.SearchControlOptions;  import com.google.gwt.search.client.SearchResultsHandler;  import com.google.gwt.search.client.WebResult;  import com.google.gwt.search.client.WebSearch;  import com.google.gwt.user.client.History;  import com.google.gwt.user.client.Timer;  import com.google.gwt.user.client.Window;  import com.google.gwt.user.client.rpc.AsyncCallback;  import com.google.gwt.user.client.ui.Anchor;  import com.google.gwt.user.client.ui.Button;  import com.google.gwt.user.client.ui.HTML;  import com.google.gwt.user.client.ui.HorizontalPanel;  import com.google.gwt.user.client.ui.Image;  import com.google.gwt.user.client.ui.Label;  import com.google.gwt.user.client.ui.SimplePanel;  import com.google.gwt.user.client.ui.TextBox;  import com.google.gwt.user.client.ui.VerticalPanel;  import edu.njit.osws.client.OntologySuggestService;  import edu.njit.osws.client.OntologySuggestServiceAsync;  import edu.njit.osws.client.OntologySuggestion;  import edu.njit.osws.client.Setting;  public class OntologySuggestBox extends VerticalPanel {    private HorizontalPanel searchPanel;  // The text box the user enters their query into  private TextBox textBox;  // The popup that that displays suggestions  SuggestionPanel popup;  // The panel that holds the suggestions  private SuggestionPanel suggestionPanel;  HorizontalPanel negativeTermPanel;  private HorizontalPanel searchButtonPanel;  // Timer that runs after a key has been pressed.  private Timer suggestTimer = new Timer() {  public void run() {  updateSuggestion();  }  };  private String savedText = "";  private SettingPanel settingPanel;  private SimplePanel topMargin;  private Image image;  private Image imageSmall;  private Button searchButton;  private Button luckyButton;    private HorizontalPanel resultTable;  private SearchControl searchControl;    private VerticalPanel inputPanel;  private HorizontalPanel input;    public OntologySuggestServiceAsync ontologySuggestService = GWT  .create(OntologySuggestService.class);  public int suggestOffset = 0;  public int suggestTotal = 0;    SearchResultsHandler defaultHandler = new SearchResultsHandler() {  public void onSearchResults(final SearchResultsEvent event) {  JsArray<? extends Result> results = event.getResults();  clearSearchResult();  VerticalPanel resultColumn = new VerticalPanel();  resultColumn.setSpacing(10);    long resultCount = event.getSearch().getCursor().getEstimatedResultCount();  Label resultLabel = new Label("About "+resultCount+" results");  resultLabel.setStyleName("resultCount");  resultColumn.add(resultLabel);    for (int i = 0; i < results.length(); i++) {  WebResult result = (WebResult) results.get(i);  resultColumn.add(result.getHtml());  // resultColumn.add(event.getSearch().getCursor());  }  resultTable.add(resultColumn);    JsArray<Page> pages = event.getSearch().getCursor().getPages();  HorizontalPanel pagePanel = new HorizontalPanel();  pagePanel.setSpacing(5);  for (int i = 0; i < pages.length(); i++){  Page p = pages.get(i);  final int index = i;  if(event.getSearch().getCursor().getCurrentPageIndex()!=i){  Anchor anchor = new Anchor(p.getLabel());  anchor.addClickHandler(new ClickHandler(){  public void onClick(ClickEvent e) {  History.newItem("page="+(index+1));  event.getSearch().gotoPage(index);  }  });  pagePanel.add(anchor);  }  else  pagePanel.add(new HTML(p.getLabel()));  }  pagePanel.add(new HTML("<a href=\"" + event.getSearch().getCursor().getMoreResultsUrl() + "\"> More results >> </a>"));  resultColumn.add(pagePanel);    }  };    public OntologySuggestBox() {  // loader and store  // BasePagingLoader<PagingLoadResult<OntologySuggestion>> loader = new BasePagingLoader<PagingLoadResult<OntologySuggestion>>(  // proxy);  // loader.load(0, 50);  // final ListStore<OntologySuggestion> store = new ListStore<OntologySuggestion>(loader);  topMargin = new SimplePanel();  topMargin.setHeight("100px");  topMargin.setWidth("100%");  add(topMargin);    setHorizontalAlignment(HorizontalPanel.ALIGN\_CENTER);  image = new Image();  image.setUrl("http://web.njit.edu/~ys89/osws/logo-new.png");  add(image);    imageSmall = new Image();  imageSmall.setUrl("http://web.njit.edu/~ys89/osws/logo-small-new.png");  imageSmall.setPixelSize(400, 48);  imageSmall.setVisible(false);  add(imageSmall);  SimplePanel divider = new SimplePanel();  divider.setHeight("5px");  divider.setWidth("100%");  add(divider);  searchPanel = new HorizontalPanel();  searchPanel.setHorizontalAlignment(ALIGN\_CENTER);    // imageSmall = new Image();  // imageSmall.setUrl("http://web.njit.edu/~ys89/osws/logo-small-new.png");  // imageSmall.setPixelSize(240, 63);  // imageSmall.setVisible(false);  // searchPanel.add(imageSmall);  // searchPanel.setCellVerticalAlignment(imageSmall, ALIGN\_TOP);  // searchPanel.setCellHorizontalAlignment(imageSmall, ALIGN\_CENTER);    inputPanel = new VerticalPanel();  input = new HorizontalPanel();  input.setVerticalAlignment(ALIGN\_MIDDLE);  inputPanel.add(input);        textBox = new TextBox();  textBox.setWidth("650px");  // textBox.setHeight("23px");  textBox.addStyleName("textboxStyle");  // textBox.setStyleName("textboxStyle");  textBox.addKeyUpHandler(new KeyUpHandler() {  public void onKeyUp(KeyUpEvent kue) {  // If its an action key  if (kue.isDownArrow() || kue.isUpArrow()  || kue.getNativeKeyCode() == KeyCodes.KEY\_ENTER) {  // And there are suggestions  if (suggestionPanel.getSuggestionCount() > 0) {  if (kue.isDownArrow()) {  suggestionPanel.moveHighlightDown();  } else if (kue.isUpArrow()) {  suggestionPanel.moveHighlightUp();  } else {  if (popup.isShowing()  && suggestionPanel.getSelectedIndex() > -1) {  String suggest = suggestionPanel  .getSelectedText();  if (suggest != null) {  closePopup();  }  } else {  doSearch(textBox.getText());  }  }  }  } else {  if (!textBox.getText().equals("")) {  // Set the timer to execute the update in 150ms  saveEnteredText();    suggestTimer.cancel();  suggestTimer.schedule(Setting.UPDATE\_TIME);  } else {  closePopup();  clearSearchResult();  }  }    }  });  popup = suggestionPanel = new SuggestionPanel(this);  // popup.add(suggestionPanel);  popup.setWidth("732px");  popup.setStyleName("suggestStyle");  input.add(textBox);  searchPanel.add(inputPanel);  add(searchPanel);  settingPanel = new SettingPanel(this);  settingPanel.addStyleName("settingPanel");  add(settingPanel);    // The search buttons  searchButton = new Button("Google Search");  searchButton.addStyleName("searchButton");  luckyButton = new Button("I'm Feeling Lucky");  searchButton.addClickHandler(new ClickHandler() {  public void onClick(ClickEvent e) {  doSearch(textBox.getText());  }  });  luckyButton.addClickHandler(new ClickHandler() {  public void onClick(ClickEvent e) {  doLuckySearch(textBox.getText());  }  });  searchButtonPanel = new HorizontalPanel();  searchButtonPanel.setHorizontalAlignment(HorizontalPanel.ALIGN\_CENTER);  searchButtonPanel.setVerticalAlignment(VerticalPanel.ALIGN\_MIDDLE);  searchButtonPanel.add(searchButton);  searchButtonPanel.add(luckyButton);  searchButtonPanel.setWidth("100%");  // add(searchButtonPanel);    resultTable = new HorizontalPanel();  resultTable.addStyleName("resultTable");  resultTable.setWidth("90%");  resultTable.setSpacing(5);  SearchControlOptions options = new SearchControlOptions();  WebSearch webSearch = new WebSearch();  options.add(webSearch);  searchControl = new SearchControl(options);  webSearch.setResultSetSize(ResultSetSize.LARGE);  searchControl.addSearchResultsHandler(defaultHandler);  add(resultTable);    setCellHorizontalAlignment(searchPanel, ALIGN\_CENTER);    VerticalPanel footer = new VerticalPanel();  footer.setHorizontalAlignment(ALIGN\_CENTER);  Label copyright = new Label();  copyright.setText("Copyright 2011 New Jersey Institute of Technology, Newark, New Jersey 07102.");  Label version = new Label();  version.setText(" Ontology-Supported Web Search version "+Setting.VERSION+".");  footer.add(copyright);  footer.add(version);  footer.addStyleName("footer");  add(footer);  setCellVerticalAlignment(footer, ALIGN\_BOTTOM);  }    // Sets the text of the query text box  public void setTextBoxText(String text) {  if (Setting.NEGATIVE\_TERMS) {  textBox.setText(text.replaceAll("\\[but not\\] ", " -").replaceAll(  "\\[and not\\] ", " -"));  } else {  textBox.setText(text);  }  }  // Saves the text that was typed by the user  public void saveEnteredText() {  this.savedText = textBox.getText();  }  // Loads the saved text into the text box  public void restoreTypedText() {  this.setTextBoxText(savedText);  }  // Closes the popup and reset it for when its opened again  public void closePopup() {  inputPanel.remove(popup);  this.suggestionPanel.resetSelectedIndex();  }  // Does a Google search on the text that has been entered  public void searchEnteredText() {  doSearch(textBox.getText());  }  // Does an "I'm Feeling Lucky" search on the text that has been entered  public void luckySearchEnteredText() {  doLuckySearch(textBox.getText());  }  // Redirects to the search results page of the given entry  public void doSearch(String entry) {  Window.Location.assign("http://www.google.com/search?q=" + entry);  }  // Redirects to the page that "I'm Feeling Lucky" results in  public void doLuckySearch(String entry) {  Window.Location.assign("http://www.google.com/search?btnI=745&q="  + entry);  }  public void updateNegativeSearchTerms(boolean value) {  if (popup.isShowing()) {  updateSuggestion();  }  }  public void giveFocus() {  textBox.setFocus(true);  textBox.setCursorPos(textBox.getText().length());  }  // Updates the suggestion panel with a new set of suggestions  public void updateSuggestion(final int offset) {  suggestOffset = offset;  final AsyncCallback<OntologySuggestion[]> getSuggestionCallback = new AsyncCallback<OntologySuggestion[]>() {  public void onSuccess(OntologySuggestion[] suggestions) {  suggestTotal = suggestions.length;    OntologySuggestBox.this.remove(topMargin);  OntologySuggestBox.this.remove(image);  OntologySuggestBox.this.remove(searchButtonPanel);  OntologySuggestBox.this.remove(settingPanel);  imageSmall.setVisible(true);      searchButton.setText("Search");  searchButton.setWidth("74px");  searchButton.setHeight(textBox.getOffsetHeight()+"px");  input.add(searchButton);    popup.setWidth(input.getOffsetWidth()+"px");    if (!popup.isShowing() && !textBox.getText().equals("")) {  inputPanel.add(popup);  } else if (textBox.getText().equals("") && popup != null) {  closePopup();  return;  }  if (suggestions.length > 0) {    suggestionPanel.updatePanel(suggestions, savedText);  } else {  closePopup();  clearSearchResult();  searchControl.execute(textBox.getText());  }        }  // Display a warning on failure  public void onFailure(Throwable caught) {  }  };  // Get the data from the servlet  ontologySuggestService.getSuggestions(savedText, Setting.NEGATIVE\_TERMS,offset,  getSuggestionCallback);  }    void updateSuggestion() {  updateSuggestion(0);  };    public SearchControl getSearchControl(){  return searchControl;  }    public HorizontalPanel getSearchResultPanel(){  return resultTable;  }    public void clearSearchResult(){  resultTable.clear();  }  } |

## SettingPanel.java

|  |
| --- |
| package edu.njit.osws.client.widget;  import com.extjs.gxt.ui.client.event.Events;  import com.extjs.gxt.ui.client.event.Listener;  import com.extjs.gxt.ui.client.event.SliderEvent;  import com.extjs.gxt.ui.client.widget.Slider;  import com.extjs.gxt.ui.client.widget.Text;  import com.google.gwt.event.dom.client.ClickEvent;  import com.google.gwt.event.dom.client.ClickHandler;  import com.google.gwt.i18n.client.NumberFormat;  import com.google.gwt.user.client.ui.CheckBox;  import com.google.gwt.user.client.ui.HorizontalPanel;  import com.google.gwt.user.client.ui.VerticalPanel;  import edu.njit.osws.client.Setting;  public class SettingPanel extends VerticalPanel {    private HorizontalPanel negativeTermPanel;  private CheckBox negativeBox;  private HorizontalPanel sliderPanel;  private Slider slider;    public SettingPanel(final OntologySuggestBox parent){  setHorizontalAlignment(ALIGN\_CENTER);  negativeTermPanel = new HorizontalPanel();  negativeTermPanel.setHorizontalAlignment(HorizontalPanel.ALIGN\_CENTER);  negativeBox = new CheckBox(" Negative Search Terms");  negativeBox.setValue(Setting.NEGATIVE\_TERMS);  negativeBox.addClickHandler(new ClickHandler() {  public void onClick(ClickEvent e) {  Setting.NEGATIVE\_TERMS = negativeBox.getValue();    parent.updateNegativeSearchTerms(negativeBox.getValue());  negativeBox.setFocus(false);  parent.giveFocus();  }  });  negativeTermPanel.add(negativeBox);  negativeTermPanel.setWidth("100%");  add(negativeTermPanel);    sliderPanel = new HorizontalPanel();  sliderPanel.setHorizontalAlignment(HorizontalPanel.ALIGN\_CENTER);  final NumberFormat formatter = NumberFormat.getFormat("0.0");  final Text time = new Text("Hover Time: "+formatter.format(Setting.HOVER\_TIME/1000.0f)+" second(s) ");  sliderPanel.add(time);  slider = new Slider();  slider.setWidth("100px");  slider.setIncrement(100);  slider.setMaxValue(5000);  slider.setMinValue(100);  slider.setValue(Setting.HOVER\_TIME);  slider.setUseTip(false);  // slider.setMessage("{0} ms");  slider.setClickToChange(false);  slider.addListener(Events.Change, new Listener<SliderEvent>() {  public void handleEvent(SliderEvent be) {  int newHoverTime = slider.getValue();  Setting.HOVER\_TIME = newHoverTime;  time.setText("Hover Time: "+ formatter.format(Setting.HOVER\_TIME/1000.0f)+" second(s) ");  }  });  sliderPanel.add(slider);  add(sliderPanel);  }  } |

## SuggestionPanel.java

|  |
| --- |
| package edu.njit.osws.client.widget;  import java.util.ArrayList;  import java.util.HashMap;  import com.extjs.gxt.ui.client.event.BaseEvent;  import com.extjs.gxt.ui.client.event.ButtonEvent;  import com.extjs.gxt.ui.client.event.Events;  import com.extjs.gxt.ui.client.event.Listener;  import com.extjs.gxt.ui.client.event.SelectionListener;  import com.extjs.gxt.ui.client.event.SliderEvent;  import com.extjs.gxt.ui.client.widget.Slider;  import com.extjs.gxt.ui.client.widget.button.Button;  import com.extjs.gxt.ui.client.widget.toolbar.FillToolItem;  import com.extjs.gxt.ui.client.widget.toolbar.LabelToolItem;  import com.extjs.gxt.ui.client.widget.toolbar.PagingToolBar.PagingToolBarImages;  import com.extjs.gxt.ui.client.widget.toolbar.SeparatorToolItem;  import com.extjs.gxt.ui.client.widget.toolbar.ToolBar;  import com.google.gwt.core.client.JsArray;  import com.google.gwt.event.dom.client.ClickEvent;  import com.google.gwt.event.dom.client.ClickHandler;  import com.google.gwt.event.dom.client.MouseOutEvent;  import com.google.gwt.event.dom.client.MouseOutHandler;  import com.google.gwt.event.dom.client.MouseOverEvent;  import com.google.gwt.event.dom.client.MouseOverHandler;  import com.google.gwt.i18n.client.NumberFormat;  import com.google.gwt.search.client.Result;  import com.google.gwt.search.client.ResultSetSize;  import com.google.gwt.search.client.SearchResultsHandler;  import com.google.gwt.search.client.WebResult;  import com.google.gwt.search.client.WebSearch;  import com.google.gwt.user.client.Timer;  import com.google.gwt.user.client.ui.HTML;  import com.google.gwt.user.client.ui.Label;  import com.google.gwt.user.client.ui.SimplePanel;  import com.google.gwt.user.client.ui.VerticalPanel;  import edu.njit.osws.client.OntologySuggestion;  import edu.njit.osws.client.Setting;  import edu.njit.osws.client.Utils;  public class SuggestionPanel extends VerticalPanel {  class MyTimer extends Timer {  private String suggest;  public void setSuggest(String s){  suggest = s;  }  public void run() {  synchronized (parent.getSearchControl()) {  parent.clearSearchResult();  parent.getSearchControl().execute(Utils.term2Query(suggest));  }  }  };    private MyTimer searchTimer = new MyTimer();    // The entries in the suggestion panel  private ArrayList<SuggestionEntry> entries;  // The suggest box that this panel is tied to  private OntologySuggestBox parent;  // The entry that is currently selected within the suggestion panel  private int selectedEntryIndex = -1;  private HashMap<String, String> searchTextMap;  public SuggestionPanel(OntologySuggestBox parentBox) {  this.entries = new ArrayList<SuggestionEntry>();  this.parent = parentBox;  searchTextMap = new HashMap<String, String>();  searchTextMap.put("plays", "music");  searchTextMap.put("hasBirthName", "birth name");  searchTextMap.put("hasBirthDate", "born on");  searchTextMap.put("hasBirthPlace", "born in");  searchTextMap.put("inTeam", "");  searchTextMap.put("inLeague", "");  }  // Updates the panel with a new set of suggestions  public void updatePanel(OntologySuggestion [] suggestions, String enteredText) {  this.clear();    // SuggestionEntry oldEntry = null;  // if(entries.size()>0)  // oldEntry = entries.get(0);    entries.clear();  parent.clearSearchResult();  int [] attributeIndex = new int[suggestions.length];  int attributeCount = 12 - suggestions.length;  int totalAttributes = 0;  for(int c = 0; c < suggestions.length; c++) {  totalAttributes += suggestions[c].getAttributeList().size();  attributeIndex[c] = suggestions[c].getAttributeList().size();  }  if(totalAttributes > attributeCount) {  attributeIndex = new int[suggestions.length];    while(attributeCount > 0) {  for(int c = 0; (c < suggestions.length) && (attributeCount > 0); c++) {  if(attributeIndex[c] < suggestions[c].getAttributeList().size()) {  attributeIndex[c]++;  attributeCount--;  }  }  }  }    int type = 1;  for(int c = 0; c < suggestions.length; c++) {  String typeStr = "type" + Integer.toString(type);  if(c > 0 && c < suggestions.length) {  // If this suggestion is of a different type  if(!suggestions[c - 1].getType().equals(suggestions[c].getType())) {  // Add a black bar  type++;  typeStr = "type" + Integer.toString(type);  add(new SuggestionDivider("black"));  } else {  // Otherwise add a colored bar  add(new SuggestionDivider(typeStr));  }  }  SuggestionEntry entry = new SuggestionEntry(suggestions[c].getFullName() + " " + suggestions[c].getType()  , enteredText, typeStr);  entries.add(entry);  add(entry);      final VerticalPanel resultColumn = new VerticalPanel();  resultColumn.addStyleName("border-"+type);  resultColumn.addStyleName("rounded-corners");  resultColumn.setHeight("100%");  resultColumn.setVerticalAlignment(ALIGN\_TOP);  resultColumn.setSpacing(10);  Label query = new Label("Google results for \""+entry.getSuggestion()+"\"");  query.setStyleName("query");  resultColumn.add(query);  parent.getSearchResultPanel().add(resultColumn);  parent.getSearchResultPanel().setCellWidth(resultColumn, 100/suggestions.length+"%");  parent.getSearchResultPanel().setCellHeight(resultColumn, "100%");  final String typeString = typeStr;    WebSearch webSearch = new WebSearch();  webSearch.setResultSetSize(ResultSetSize.LARGE);  webSearch.addSearchResultsHandler(new SearchResultsHandler() {  public void onSearchResults(SearchResultsEvent event) {  JsArray<? extends Result> results = event.getResults();  // VerticalPanel resultColumn = new VerticalPanel();  long resultCount = event.getSearch().getCursor().getEstimatedResultCount();  Label resultLabel = new Label("About "+resultCount+" results");  resultLabel.setStyleName("resultCount");  resultColumn.add(resultLabel);  resultColumn.add(new SuggestionDivider(typeString));    for (int i = 0; i < results.length(); i++) {  WebResult result = (WebResult) results.get(i);  resultColumn.add(result.getHtml());  // resultColumn.add(event.getSearch().getCursor());  }    if(event.getSearch().getCursor().getCurrentPageIndex()!=2)  event.getSearch().gotoPage(2);      }  });  webSearch.execute(Utils.term2Query(entry.getSuggestion()));        ArrayList<String> attributes = suggestions[c].getAttributeList();      for(int i = 0; i < attributeIndex[c]; i++) {  String relationship = attributes.get(i);  String rel = relationship.substring(0, relationship.indexOf(" "));  String target = relationship.substring(relationship.indexOf(" "));  entry = new SuggestionEntry(suggestions[c].getFullName() + " " +  searchTextMap.get(rel) + target, enteredText, typeStr);  add(entry);  entries.add(entry);  }  }      final ToolBar toolBar = new ToolBar();  PagingToolBarImages images = new PagingToolBarImages();  final Button prev = new Button();  prev.addSelectionListener(new SelectionListener<ButtonEvent>() {  @Override  public void componentSelected(ButtonEvent ce) {  prev.disable();  int i = parent.suggestOffset-Setting.TERMS\_UNIT;  parent.updateSuggestion(i);  }  });  if(parent.suggestOffset==0){  prev.setIcon(images.getPrevDisabled());  prev.disable();  }  else {  prev.setIcon(images.getPrev());  prev.enable();  }    toolBar.add(prev);  final Button next = new Button();  next.addSelectionListener(new SelectionListener<ButtonEvent>() {  @Override  public void componentSelected(ButtonEvent ce) {  next.disable();  int i = parent.suggestOffset+Setting.TERMS\_UNIT;  parent.updateSuggestion(i);  }  });    if(parent.suggestOffset==Setting.TERMS\_MAX-Setting.TERMS\_UNIT || suggestions.length<Setting.TERMS\_UNIT){  next.setIcon(images.getNextDisabled());  next.disable();  }  else {  next.setIcon(images.getNext());  next.enable();  }    toolBar.add(next);  toolBar.add(new SeparatorToolItem());    final com.extjs.gxt.ui.client.widget.form.CheckBox box = new com.extjs.gxt.ui.client.widget.form.CheckBox();  box.setBoxLabel("Negative Search Term");  box.setValue(Setting.NEGATIVE\_TERMS);  box.addListener(Events.OnClick, new Listener<BaseEvent>() {  public void handleEvent(BaseEvent be) {  Setting.NEGATIVE\_TERMS = box.getValue();  parent.updateNegativeSearchTerms(box.getValue());  // box.setFocus(false);  parent.giveFocus();  }  });  toolBar.add(box);  toolBar.add(new SeparatorToolItem());  toolBar.add(new LabelToolItem("Hover Time: "));  final NumberFormat formatter = NumberFormat.getFormat("0.0");  final LabelToolItem time = new LabelToolItem(formatter.format(Setting.HOVER\_TIME/1000.0f)+" second(s) ");  toolBar.add(time);  final Slider slider = new Slider();  slider.setWidth("100px");  slider.setIncrement(100);  slider.setMaxValue(5000);  slider.setMinValue(100);  slider.setValue(Setting.HOVER\_TIME);  slider.setUseTip(false);  // slider.setMessage("{0} ms");  slider.setClickToChange(false);  slider.addListener(Events.Change, new Listener<SliderEvent>() {  public void handleEvent(SliderEvent be) {  int newHoverTime = slider.getValue();  Setting.HOVER\_TIME = newHoverTime;  time.setLabel(formatter.format(Setting.HOVER\_TIME/1000.0f)+" second(s) ");  }  });  toolBar.add(slider);  toolBar.add(new LabelToolItem(""));  toolBar.add(new SeparatorToolItem());      toolBar.add(new FillToolItem());  int to = suggestions.length<Setting.TERMS\_UNIT?(parent.suggestOffset+suggestions.length):(parent.suggestOffset+Setting.TERMS\_UNIT);  toolBar.add(new LabelToolItem("Showing "+(parent.suggestOffset+1)+" - "+to));  add(toolBar);    }  // Moves the selected highlight box to the entry below it, if one exists  public void moveHighlightDown() {  selectedEntryIndex++;  if( selectedEntryIndex > (entries.size() - 1) ) {  parent.restoreTypedText();  selectedEntryIndex = -1;  unhighlightAll();  } else if( selectedEntryIndex == 0 ) {  parent.setTextBoxText(getSelectedText());  highlightEntry(entries.get(selectedEntryIndex));  } else {  parent.setTextBoxText(getSelectedText());  highlightEntry(entries.get(selectedEntryIndex));  }  }  // Moves the seelected highlight box up to the entry above the current  public void moveHighlightUp() {  selectedEntryIndex--;  if(selectedEntryIndex < -1) {  selectedEntryIndex = entries.size() - 1;  parent.setTextBoxText(getSelectedText());  highlightEntry(entries.get(selectedEntryIndex));  } else if(selectedEntryIndex == -1) {  unhighlightAll();  parent.restoreTypedText();  } else {  parent.setTextBoxText(getSelectedText());  highlightEntry(entries.get(selectedEntryIndex));  }  }  // Returns the entry that is currently selected, returns null if non selected  public String getSelectedText() {  if(selectedEntryIndex >= 0 && selectedEntryIndex < entries.size()) {  return entries.get(selectedEntryIndex).getSuggestion().toLowerCase();  } else {  return null;  }  }  // Returns the current number of suggestions  public int getSuggestionCount() {  return entries.size();  }  // Makes the given entry the only highlighted entry  private void highlightEntry(SuggestionEntry entry) {  unhighlightAll();  entry.removeStyleName(entry.getStyle() + "-normal");  entry.addStyleName("highlight");  entry.setHighlighted(true);    searchTimer.cancel();  searchTimer.setSuggest(entry.getSuggestion());  searchTimer.schedule(Setting.HOVER\_TIME);  }  // Removes the highlight from all entries  private void unhighlightAll() {  for (SuggestionEntry s : entries) {  if (s.isHighlighted()) {  unhighlight(s);  }  }  }  // Unhighlights the given entry  private void unhighlight(SuggestionEntry s) {  s.setHighlighted(false);  s.removeStyleName("highlight");  s.addStyleName(s.getStyle() + "-normal");  }  // Returnst the index of the given entry  private int getIndexOf(SuggestionEntry s) {  for(int c = 0; c < entries.size(); c++) {  if(entries.get(c) == s) {  return c;  }  }  return -1;  }  // Returns the selected index  public int getSelectedIndex() {  return selectedEntryIndex;  }  // Resets the selected index to nothing selected (-1)  public void resetSelectedIndex() {  this.selectedEntryIndex = -1;  }  // Class representing a suggestion from the ontology  public class SuggestionEntry extends SimplePanel implements MouseOverHandler, MouseOutHandler {  private String suggestion;  private HTML suggestionText;  private String style;  private boolean highlighted = false;  public SuggestionEntry(final String suggestion, String enteredText, String style) {  this.suggestion = suggestion;  this.style = style;  setWidth("100%");  enteredText = enteredText.trim();  String [] splitEnteredText = enteredText.split(" ");  String alternativeEntered = "";  if(splitEnteredText.length > 2) {  alternativeEntered = splitEnteredText[0] + " " + splitEnteredText[1];  }  String start = "";  String filledIn = "";  if (suggestion.toLowerCase().startsWith(enteredText.toLowerCase())) {  start = enteredText.toLowerCase();  filledIn = suggestion.toLowerCase().replaceFirst(start, "");  } else if (!alternativeEntered.equals("") &&  suggestion.startsWith(alternativeEntered.toLowerCase())) {  start = alternativeEntered.toLowerCase();  filledIn = suggestion.toLowerCase().replaceFirst(start, "");  }  suggestionText = new HTML(start + "<b>" + filledIn.replaceAll("\\[",  "<font color='RED' size=\"X-small\">[").replaceAll("\\]", "]</font>") + "</b>");  suggestionText.setStyleName("suggestionTextStyle");  suggestionText.addMouseOverHandler(this);  suggestionText.addMouseOutHandler(this);  suggestionText.addClickHandler(new ClickHandler() {  public void onClick(ClickEvent ce) {  parent.setTextBoxText(suggestion.toLowerCase());  parent.closePopup();  }  });  add(suggestionText);  addStyleName("suggestionPanelStyle");  addStyleName(style + "-normal");  }  public String getSuggestion() {  return suggestion;  }  public String getStyle() {  return style;  }  public boolean isHighlighted() {  return highlighted;  }  public void setHighlighted(boolean value) {  this.highlighted = value;  }  public void onMouseOver(MouseOverEvent moe) {  highlightEntry(this);  selectedEntryIndex = getIndexOf(this);  }  public void onMouseOut(MouseOutEvent moe) {  unhighlight(this);  selectedEntryIndex = -1;  searchTimer.cancel();  }  }  // Class representing a divider within the suggestion list  private class SuggestionDivider extends SimplePanel {  public SuggestionDivider(String style) {  setHeight("1px");  setWidth("100%");  setStyleName("divider-" + style);  }  }  public boolean isShowing() {  return isAttached();  }  } |