WikiGraph

Mark Jordan, Jeremy Lenz, Robert McClure,
Austin Nakamura, Michael Rush, Khanh Tran,
Thomas Van Doren

Final Release

Draft 1
10 March 2011

WikiGraph
Mark Jordan, Jeremy Lenz, Robert McClure, Austin Nakamura, Michael Rush, Khanh Tran, Thomas Van Doren

CSE 403 - CSRocks Inc.

Version	Primary Author(s)	Description of Version	Date Completed
1	Thomas Van Doren	Started introduction and added a known issue	2011-03-09
2	Rob McClure	Added the section on extensibility	2011-03-09
3	Thomas Van Doren	Copied the lessons from the presentation	2011-03-10
4	Rob McClure	Added the section on design patterns	2011-03-10
5			

WikiGraph
Mark Jordan, Jeremy Lenz, Robert McClure, Austin Nakamura, Michael Rush, Khanh Tran, Thomas Van Doren

Introduction

The final release is live, and the WikiGraph team couldn't be happier! Direct your browser to:

http://wikigraph.cs.washington.edu/

We are pleased to include all of the features we committed to and even a few stretch features! We have cleaned up the user interface by adding some element stlying and resizing. The graph includes some animation to keep the user interested, as well as keeping a history of viewed graphs. Every graph has a unique url so that it can easily be revisited. Graphs now display inbound and outbound connections between articles. Double clicking on graph nodes opens the articles

The services and database have also been updated. Search is now case insensitive and returns results using the soundex algorithm (phrases that sound alike). Search will also automatically return a graph if a close match was found. The graphs also are cached for faster performance. When a graph is viewed, first the cache is examined. If it is not already cached, a mix of inbound and outbound links are quickly returned to the user and a background job is initiated to cache the graph. The cache is done as a background job because it can take up to 30 seconds for some graphs.

We are very happy with our release and hope that many users will find it useful and interesting.

Design Patterns

The WikiGraph team does not have a strict coherence to any large scale pattern (like those in the Gang of Four). However, in the flash client, and in some of the OO PHP scripts, data encapsulation is used to protect the state of objects and encourage appropriate usage. One file that this is particularly noticeable is the Parse class (`client/FlexClient/DrawGraph/src/Parse.as`).

In the parse class, only the four method which actually interact with other entities in the system are public. The helper method is private such that it cannot be used unintentionally by naive (or malicious) programmers. The fields of the Parse class are not accessible outside of the class. By encapsulating these values, Parse has greater control over their lifetime. The internal intermediate state of Parse is inconsequential to outside entities. Therefore, there is no need to allow access to these parameters.

Additionally, we use a facade to interact with our database. The database contains a number of tables with fields that connect in certain ways. However, we don't want the services to have to deal with the details of connecting to the database and sending queries. As a result, we created a GraphDB class (in `services/util.php`) which takes care of connecting to the database and forming the correct queries. The services can use this class to get the information from the database (such as getting page information or page links) without knowing how or where the data is stored in the database.

Finally, we use an adapter in the form of our database schema. Through a series of transformations (found in `database/transform.sql`), we take the schema from the wikipedia dumps (or other MediaWiki dump) and convert it to our schema. Both the original schema and our schema have the same information, but it is grouped together in different ways. Thus, in a sense the functionality is the same, but the interface has been changed.

Extensibility

Our services rely on interacting with a specific database schema, which is outlined in our Software Development Specification. In order to use an online encyclopedia with a different database schema, we would need to transform from their schema to our schema. In fact, we already do this with the Wikipedia (and other MediaWiki) databases. An example of this transformation can be found in our transform.sql file, which transforms a MediaWiki schema to our schema. Depending on how different the encyclopedia schema is from the MediaWiki schema, this SQL script would need to be altered to deal with the different table and field names. All our schema relies on is having a set of pages (in the page table), the links between the pages (the pagelinks table), and abstracts for the pages (the abstract table). If this information can be found in the encyclopedia schema, then it can easily be transformed to our schema. In fact, not all of the fields in our schema need to be present; for example, if the encyclopedia doesn't have ambiguous pages, then the page_is_ambiguous field in our page table could be left at a default value.
As a result, using a different schema is simply a matter of transforming it to our schema. In our

example transform.sql, there are comments about what may need to be changed to make the transformation work without writing a separate transform script.

Top Three Lessons Learned

- Large team dynamics
    - Communication
    - Time management
- Flash development
    - Open source
    - Easy to learn
    - Optimized performance
- Distributed source control
    - Merging
    - Conflicts

Known Issues

- Back/Forward browser buttons do not work - this was a stretch feature
- In some situations the links for a node are cached multiple times