Journal of Scientific and Practical Computing
Vol. 2, No. 1 (2008) 21–29

DEVELOPMENT OF A WEB-BASED APPLICATION TO GEOGRAPHICALLY PLOT WATER QUALITY DATA

TRAVIS L. BRANHAM

Department of Computer Science and Information Technology, The University of the District of Columbia, 4200 Connecticut Ave. NW, Washington, DC 20008, USA

Abstract:

Government agencies, such as the U.S. Geological Survey, provide vast quantities of water quality data to the public, but it is often difficult for a citizen to find the data relevant to their specific location. Using a web-based application framework, along with Google Maps technology, it is possible to build a site that provides a geographical reference to the water quality data, allowing citizens to easily access information. The goal of this application is to bridge the gap between the availability of the data and its accessibility. Citizens visiting the site will be given the ability to search for the water quality testing locations nearest to them, and will then be presented with the actual testing results that have been found for that location on any given test date.

Keywords: Geo-location

1. Introduction

A vast quantity of information, on many subjects that may be of interest to the general public, is offered, through various government agencies, free of charge; the methods for acquiring and understanding this data, however, can be difficult and troublesome for many individuals. The web-based application outlined in this paper attempts to address the difficulties inherent in the dissemination of data to interested members of the general public, and specifically data which has a geographical reference component.

This application deals specifically with water quality data in the metropolitan Washington, D.C. area, as reported by the U.S. Geological Survey (USGS). Interested individuals will access the web-based utility and, from there, easily navigate to the desired information. The data is presented initially by icons, representing water quality testing locations, plotted onto a map of the metropolitan Washington, D.C. region; as the user clicks an icon, he or she will be presented with a list of dates, with each date corresponding to an actual water quality test. By clicking on a date, users will then be presented with a page-by-page view of the values for each type of test performed at that time (e.g. lead, arsenic, etc.). In addition to the ability to browse these sites randomly, users may enter an address, and a radius, to find only the sites of closest geographical interest.

The application, in its current form, can be found at: http: //www.travisbranham.com /waterquality/

2. Background and Related Work

The impetus for building this application came, in part, due to research that the author performed while working with Dr. Li Chen (University of the District of Columbia) and the Water Resource Research Institute (WRRI) of the University of the District of Columbia, involving water quality data for the Washington, D.C. area, which was extracted from the USGS National Water Information System Web Interface[1]. The data from the USGS was presented in several flat, tab-delimited text files (some embedded within one another), corresponding to 4 main categories: information related to the testing locations, the test parameters, the test results, and the various applicable remarks to those results. After the files were downloaded, and certain formatting modifications were performed, they were loaded into a MySQL database for easy access. The work for Dr. Chen consisted of pulling certain types of records, based on specific date and location criteria, and formatting the output so it could be used by other applications.

As the work progressed, an opportunity arose to conduct some independent research for the Louis Stokes Alliance for Minority Participation (LSAMP) Summer Research Institute. The topic chosen was to develop a computer application, which would simplify the accessing of the USGS water quality data for residents in the metropolitan D.C. area.

There were three main objectives that the application would have to achieve:

(1) The application should be web-accessible.
(2) The relevant data should be displayed on a map.
(3) Access to the information should be fast and efficient.

Research was conducted in these three areas in order to find projects with similar design considerations. Ruby on Rails was chosen as the web application framework, due to its flexibility and ease-of-use, and the Google Maps Application Programming Interface (API) was chosen as the mapping interface.

3. Application Overview

The application has four main components:

(1) The web application framework (Ruby on Rails).
(2) The database (MySQL).
(3) The mapping component (the Google Maps API).
(4) Asynchronous JavaScript and XML (Ajax).

Each of these components is discussed in detail in the following subsections.

3.1 Ruby on Rails

Ruby on Rails (RoR) is a web application framework, built on top of an object-oriented programming language known as Ruby. RoR uses a model-view-controller paradigm[2] to help separate the different parts of a database-backed web application from one another.

In the model, the programmer defines a set of objects, each corresponding to a database table. These objects work best when the contents of the database table represent a single classification of things, such as a table for 'sites,' which only stores information related to the testing locations, and not to the values acquired at those locations. In short,

the objects work with well-designed database tables. The advantage to this relationship is that RoR transparently generates methods for the user-defined objects to facilitate easy database access.

A RoR view is a set of code, written in a markup language called RHTML, which allows for the execution of Ruby code to generate parts of the displayed web page. This code allows dynamic content, from sources such as a database, to be displayed based on user input. In addition, other elements may be added to the view, such as JavaScript code or static HTML content.

```
Example: RHTML format:
      Code executed without output:
      <% #code goes here %>
      Code executed with output:
      <%= #code goes here %>
```

Finally, a RoR controller is used to communicate between the different elements of a dynamic web application, and to generate responses to user input. The controller, in this case, acts as a bridge between the database model and the users web browser. Additionally, the controller is responsible for communicating with outside libraries, such as the Google Maps API or the Prototype Ajax library.

3.2 MySQL

The original data extracted from the USGS website was stored in a MySQL database. Since the Ruby on Rails framework already has a tight integration with MySQL, it was decided to maintain the data in this format. There was, however, some data stored in the tables that was not required for this application, and was therefore removed to conserve storage space.

In order to benefit from the automatic code generation features from the Rails framework[2], some of the table columns were required to follow a set of predefined naming conventions dealing with pluralization. These conventions allow for a table to contain information on 'sites', while the object represents access to a single 'site'.

The database tables are laid out in such a way as to encapsulate the primary elements needed to describe the various aspects of each test and testing location. The Rails framework expects the unique id for each table to be called 'id'; subsequent fields may be named as appropriate for the application. Additionally, table relationships are facilitated by adding fields to each table, which map to the unique identifier of another table. For example, by adding a field to the 'results' table called 'parameter_id', it is possible to lookup the associated parameter information when dealing with a 'Result' object. Figure 1 illustrates the database schema, which fulfills all of the basic application requirements.

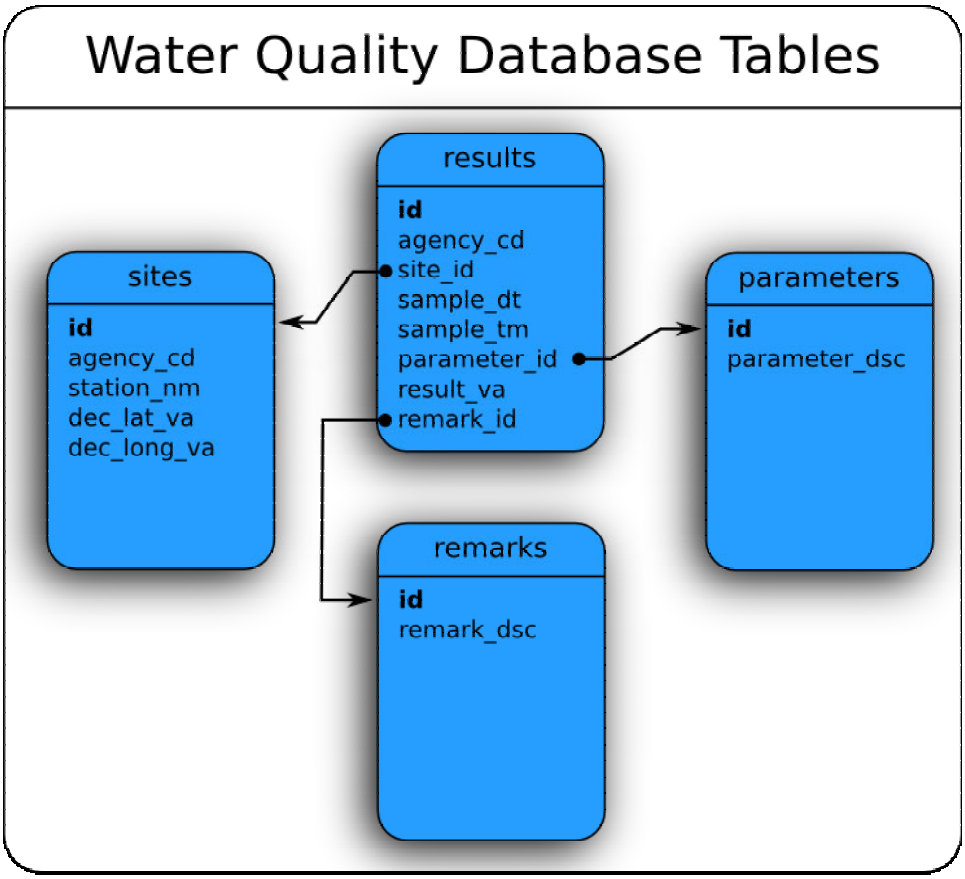

Figure 1: An example of the database schema.

3.3 The Google Maps API

The Google Maps API[3] was chosen for this project for two reasons: it offers a simple and convenient method for user interaction, and it provides a geocoding service[4], which translates a user-defined address (such as a street address) into a set of latitude and longitude coordinates.

Interfacing with the Google Maps API (Version 2) is a matter of using a series of JavaScript functions, in combination with a Google API Key, to initialize the various mapping components of the display.

After initializing the map, there are other JavaScript functions, which allow the developer to add custom icons and text-boxes to the map. For instance, the USGS data provides a latitude and longitude reference for each of the testing sites; by writing a routine in Ruby, one can add a point on the map for each of the testing sites by placing the coordinates into the appropriate JavaScript functions.

```
Example: Plotting Points with the Google Maps API:
      <% @sites.each do |site| %>
      var point = new GlatLng(site.dec_lat_va,
      site.dec_long_va);
      var marker = new createMarker(point);
      map.addOverlay(marker);
      <% end %>
```

This example does not address the additional HTML content that is passed to each marker, which lists some general information about each site. Figure 2 details the position and content of the marker overlays. Updates and additions to the map are made using similar sets of JavaScript functions in concert with embedded Ruby code.

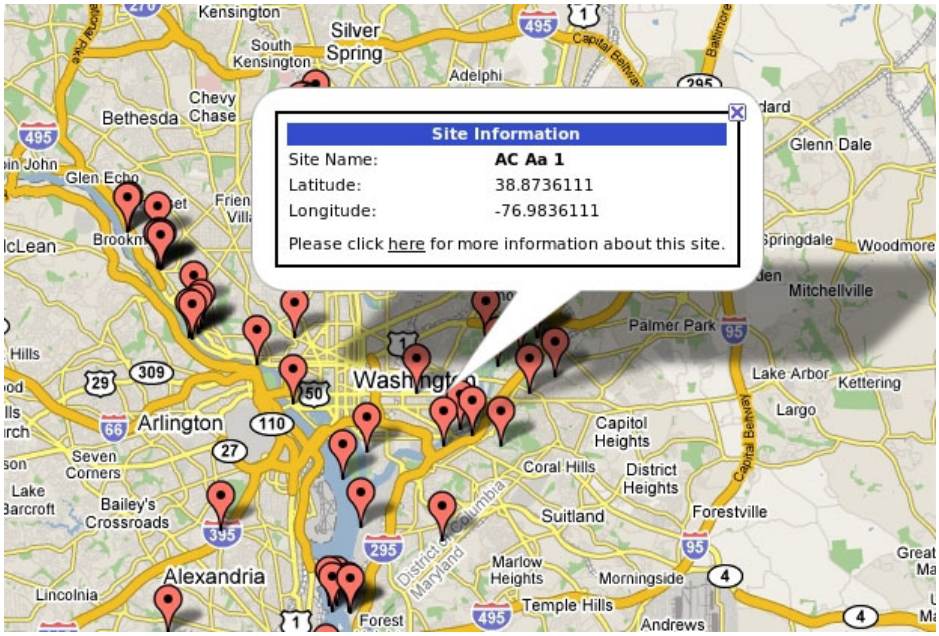

Figure 2: Detailed view of the marker overlays. Clicking on a marker reveals detailed site information.

A Ruby extension called GeoKit is used to better facilitate communication between the application and the geocoding service provided by Google. This module simplifies the task of geocoding user-supplied addresses, as well as distance calculation between locations.

3.4 Ajax

Ajax describes a method of communication between the client and the server in an asynchronous manner. This simply means that clicking a link does not necessarily result in an entire page load; rather, targeted portions of the page may be updated, while other parts remain static. This method improves access speeds for web applications, because

the entire application does not generally lose functionality during a bandwidth-expensive load.

Ruby on Rails has built-in integration with several Ajax libraries, including Prototype and Script.aculo.us, which makes incorporating Ajax elements relatively painless. A simple, one-line command is enough to add the libraries to the current project. After the libraries have been added, one has only to use the built-in Ruby commands to activate specific Ajax elements.

This application makes extensive use of the Ajax-enabled 'form_remote_tag' method, which sends an 'XMLHttpRequest' between the client (in this case, the user's web browser) and the web server in response to a web-form submit action. This method allows the server to update the targeted HTML element without causing the entire page to reload.

4. The Complete Application

After building and testing the development version of the application, the finished components are encapsulated within a Ruby on Rails project and placed on the web server. Additionally, the MySQL database on the web server is populated with the information from the USGS online database. Figure 3 gives an overall schematic of the entire application.

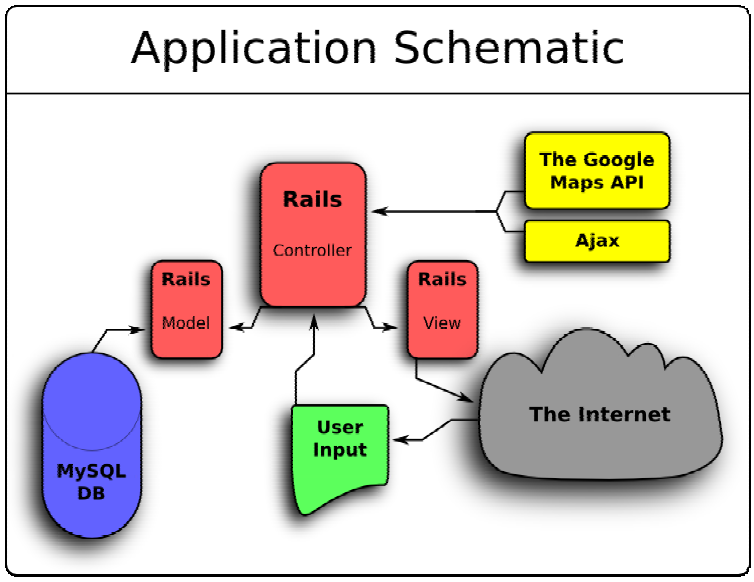

Figure 3: The complete application.

At the start of a typical session, the first action is to initialize the Google Maps interface. After the map has been initialized, the database is queried, by using an object method call, and an icon is added to the map for each of the water testing sites. In addition, a web form element (see Figure 4) is created in RHTML to accept an address and a distance value from the user.

To help prevent SQL injection attacks[a], or other malicious behavior, the user input is never allowed to reach a position where it can be directly executed by the Ruby programming language (additionally, the distance value is provided from a drop-down menu). The address is sent to the GeoKit class[4] for geocoding, and a GeoKit object is created which contains the information derived from the geocoding service provided by Google.

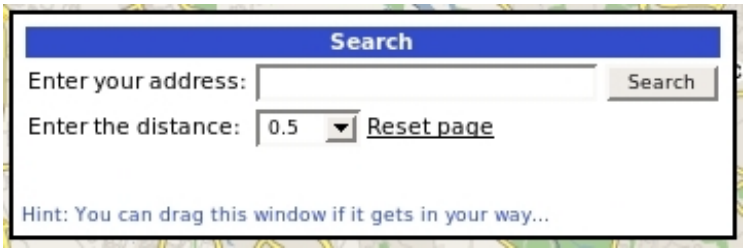

Figure 4: The search box.

If there is a positive match, the database is then queried by GeoKit to find the sites which fall within the defined radius; once found, the map is cleared of the existing icons, and the set of icons within the region of interest are placed back on the map (with the addition of an icon corresponding to the user's location).

By clicking on any of the icons, which represent a testing location, a pop-up window appears on the map, indicating the name and geographical coordinates of the location (see Figure 5). This window also provides a link to the actual testing results for that particular site.

Clicking on the information link will bring up another pop-up window, using Ajax, which indicates the dates which water from that site had been tested on. Once a date has been selected, the window will refresh with a page-by-page view of each parameter tested at that site on that particular day.

[a] Executable code given to the interface by a user, with the intent of exposing information from the underlying database.

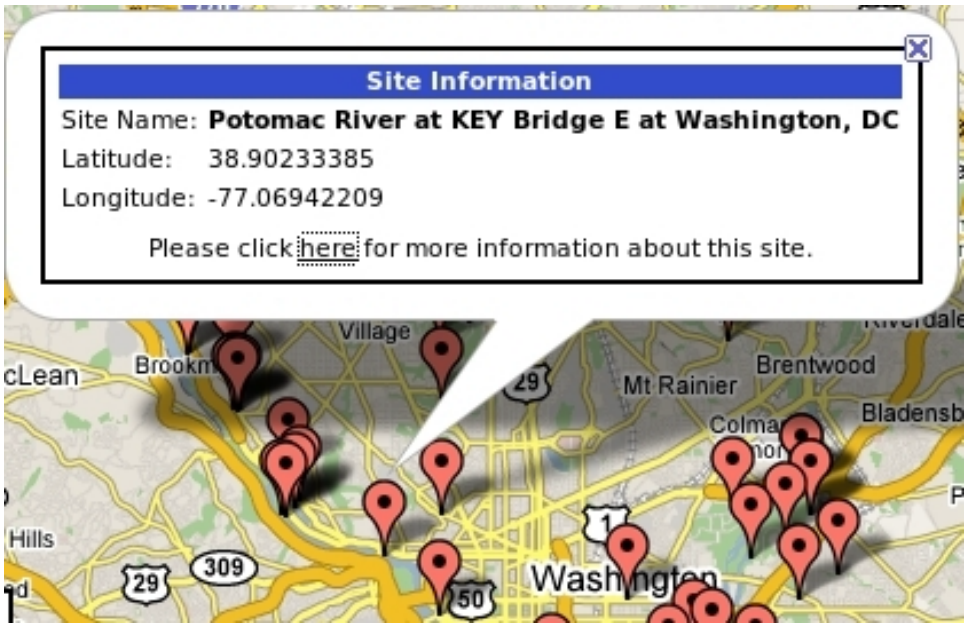

Figure 5: A link to the testing information for a particular site.

5. Future Work

There are several additions, which would add a tremendous amount of functionality to the current site:

1. The addition of useful links to groups, such as the Environmental Protection Agency, related to water quality and conservation issues.
2. A comprehensive data clean-up project. For example, the current database lists parameter names with the associated units as part of the text (e.g. “Temperature, Water, degrees Celsius”); it would be preferable to store the units in a separate text field.
3. A revision and cleanup of the current Cascading Style Sheets (CSS).
4. The incorporation of useful tools, which can assist the blind in accessing the information provided by the site in a clear and consistent manner.
5. Building a “print-friendly” formatting scheme to allow individuals to make hard copies of the data they find on the site.

This application also relies on a static dump of the USGS database. This is obviously not a scalable solution, as it requires a large amount of manual work to add new information into the database. The application, while tailored to work with the USGS data, could easily be modified to work with data from other sources. In fact, the ideal situation is to modify the Ruby on Rails model to accept a wider range of data, and write a simple script to manipulate whatever source data is available into a form suitable for database upload.

6. Conclusions

In an application such as the one outlined in this paper, it is difficult to quantitatively assess whether or not the application satisfies the initial design goals. However, a qualitative analysis suggests that the application simplifies the access of water quality information for residents of the metropolitan D.C. area by providing a simple web-based map.

Simply facilitating access to this information is not enough, however, to determine the success of the project either; the USGS data was already freely available prior to designing the application[1]. To consider the application truly successful, a deeper understanding of the information should be achieved by individuals accessing the application.

Astute observers will note, after viewing the site, that there is a paucity of water quality information available for the metropolitan D.C. area from the USGS. To achieve maximum effectiveness, other sources of water quality data should be pursued to supplement geographical and chronological gaps.

Acknowledgments

The author would like to thank William W. Hare (UDC, WRRI), Mary Farrah (WRRI), Dr. Pradeep K. Behera (UDC – Dept. Civil Engineering), Dr. Esther T. Ososanya (UDC – Dept. Electrical Engineering, LSAMP), and the numerous individuals who frequent the online forums for the Google Maps API and Ruby on Rails development.

References

1. U.S. Department of the Interior. USGS Water Data for USA. U.S. Geological Survey. http://waterdata.usgs.gov/nwis
2. Tate, B. A., Hibbs, C. 2006 Ruby on Rails, Up and Running. O'reilly Media, Inc.
3. Google, Inc. 2008. Google Maps API Reference. http://code.google.com/apis/maps/documentation/reference.html
4. Eisenhauer, B., Lewis, A. 2007. GeoKit for Rails, A Rails plugin for easier map-based applications. http://geokit.rubyforge.org/