University of North Carolina at Asheville
Journal of Undergraduate Research
Asheville, North Carolina
May 2013

Web Application Development for Scientific Data

Nathaniel Bennett, Rebekah David, Emily Watson
Computer Science
The University of North Carolina at Asheville
One University Heights
Asheville, North Carolina 28804 USA

Faculty Advisor: Susan Reiser

Abstract

Scientific data is often not very useful in its raw formats, especially in discovery-based science, where hypotheses are formed by analyzing large multivariate datasets. Scientists who work on experimental data need to formulate context around the datasets they are analyzing. Web application frameworks are tools for creating context, visualizing and facilitating collaboration needed to understand many different types of commercial and analytical information. This research will entail finding a suitable web application framework, then designing and developing a system for Dr. Airat Khasanov, a chemistry research scientist at UNC Asheville. This research lends itself well to future students as well as the broader scientific community. It also offers insight into how conventional, open and easily accessible web tools can be used to assist data-driven science in forming hypotheses. Furthermore, it may serve as a template for further interdisciplinary collaboration.

1. Introduction

A web application is specifically designed and programmed to be rendered by web browser packages. These operating system independent applications use Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript for their user interfaces. In computer science, HTML and CSS are well-documented methods of space description, structuring and formatting a document to be displayed in a web browser. JavaScript is a programming language that provides programmers the ability to add automation, interactivity and dynamism for users. In the context of a web browser, these three well documented, client side languages can create and constitute a web application.

However, there are limitations to the range of purposes that client side only web applications can serve. Adding server side programming greatly extends these capabilities by enabling the application to store vast amounts of data, provide more dynamism and permit the user to manipulate and interact with data. Server side programming languages commonly utilized in web application development include Python, PHP and Perl. These three languages have vast capabilities on web servers because they specialize in scripting operations, which automate the methods of transferring information between a database and an HTML-formatted file, so that the information transfer may occur in a certain order or in response to user interaction. Developers working with both server side and client side programming often utilize web application frameworks to develop more efficient and robust applications. Frameworks can include code libraries, application programming interfaces (API) for those libraries, and templates to streamline the process of developing a complex application by providing commonly used code structures to the programmer for activities like database access and session management.

In contrast to client side only web applications, web applications that utilize a web application framework can process, store and perform calculations on data. Client side web applications may only describe and display data. Scientists who work with various types of data often seek to centralize storage, display, and process that data into useful datasets. This research project focuses on the potential of web applications to improve the accessibility and organization of scientific data, as well as to provide innovative tools for manipulating and interpreting this data.

The primary goal of the project was to find an appropriate framework and develop a web application for Dr. Airat Khasanov, a chemistry research scientist at the University of North Carolina at Asheville. Dr. Khasanov's field is considered discovery-based science and requires analysis of large multivariate datasets. This project was designed so that a prototype web application could bring context to datasets collected by Dr. Khasanov, and eventually to enable him and other scientists working in the same field to share, compare and analyze data between laboratories.

2. Background

Originally, spectroscopy referred to the use of a prism to disperse light into its component colors. Over time, that definition has evolved to NASA's definition: "...the study of spectral lines from different atoms and molecules."[1] Spectroscopy applications exist across many scientific fields including physics, chemistry, biology and astronomy and produce data that is often represented by a spectrum, which is a plot of the response as a function of wavelength or frequency. Mössbauer Spectroscopy is a specific spectroscopy technique that has played an important role in UNC Asheville's Chemistry Department since the 1970s. Professor John Stevens established the Mössbauer Effect Data Center at UNC Asheville and published the first issue of *Mössbauer Effect Reference and Data Journal*.[2] Currently, databases such as Data Archive at Mer Moessbauer Spectrometer[3] and Mount Holyoke's Mars Mineral Spectroscopy Database[4] provide Mössbauer spectrum repositories. A general spectral database that is ubiquitous in college library links is SDBS. SDBS development began in Japan in 1982, and today it is the foremost repository of organic compound spectra.[5] While the information contained in these existing web accessible databases is valuable, the spectra are displayed statically. The websites are neither responsive nor do they not allow users to input datasets. Therefore, they are only usable as searchable references.

During our project team's initial concept and planning meetings, Dr. Khasanov explained the need for a more dynamic and interactive means of handling and interpreting this spectroscopy data. Although the community of scientists working with this data worldwide is small, less than a few thousand, enhancing their ability to manage and share large datasets and examine minute changes in multivariate graphs will aid their collaboration.

3. Methods

Dr. Khasanov provided us with several sample raw datasets from his equipment. We analyzed these raw datasets and found that the files generated by his equipment are formatted in very simple, accessible formats. The desktop application that Dr. Khasanov uses to produce these files outputs plain text files. These file formats are either tab-separated or formatted using standard Extensible Markup Language (XML), which is conveniently displayed in a readable format in a web browser. Our focus for the project was to make this data useful by creating a simple way for scientists like Dr. Khasanov to upload the raw data to the web application and immediately view it as an interactive graph. The scientists would then be able to interact with the visualized data by manipulating the graph, comparing multiple visualized datasets and saving their work for later reference. Throughout the development process, we utilized Dr. Khasanov's scientific knowledge and guidance to keep the functionality streamlined and to ensure the efficacy of the application in meeting the needs of the scientific community.

To determine how to begin developing a web application for this scientific data, we had to analyze and understand the file formats that Dr. Khasanov provided. We came to understand the fundamental significance of the data by interviewing Dr. Khasanov about what the raw outputs were to eventually represent. Many of the underlying scientific concepts involved rather complex understanding of certain aspects of spectroscopy and graph theory, so Dr. Khasanov also gave our group several "crash courses" to provide us with the relevant background knowledge. We faced the challenge of learning these new concepts and then formulating ways to integrate the concepts into our design and programming process. The process of interviewing Dr. Khasanov and participating in the "crash courses" involved fluid discussion, extensive white boarding, and collaborative note taking.

3.1 Choosing Frameworks and Libraries

After interviewing Dr. Khasanov, we decided to use the CodeIgniter web application framework. CodeIgniter is a free, easy-to-install, extensively documented, open-source framework that includes a server-side programming component. It uses PHP as its scripting language and utilizes a standard development pattern known as model-view-controller.[6] Access to the model-view-controller pattern of programming involves writing scripts, called "controllers,"

which change "views" of data models. The pattern also contains data "models," which are defined by the programmer. When the data models are updated, a controller is notified to update the view. A view is the representation of the data model and the final output rendered to the user. In web applications, the view appears within the context of the web browser. Additionally, by choosing CodeIgniter, we had access to a sizable open source code base. During our first meetings, Dr. Khasanov expressed an interest in the open-source movement and encouraged us to develop our project as open-source so it can later be extended and modified by other programmers.

Even though we had an application framework, we also required a framework for space description and displaying output from the web application. To display the web application's output, we used Twitter's open source Bootstrap framework for the HTML, CSS and JavaScript view templates, which CodeIgniter updates. Having devised an application framework and a templating system for the views generated by the application framework, we needed methods to generate views that properly represented Dr. Khasanov's datasets. Multigraph is a program developed at UNC Asheville, written in JavaScript which produces graphs from specifically formatted XML. [7] The graphs that the program generates have interactive properties for which Dr. Khasanov expressed a desire. Namely, the Multigraph software can pan, zoom, scale, and

```
    <renderer type="line">
      <option name="linecolor" value="orange"/>
      <option name="linewidth" value="2"/>
      <option name="missingvalue" value="-9999.9999"/>
      <option name="missingop" value="EQ"/>
    </renderer>
  </plot>
  <plot>
    <legend label="Mean Dew Point"/>
    <horizontalaxis ref="haxis">
      <variable ref="time"/>
    </horizontalaxis>
    <verticalaxis ref="dewp">
      <variable ref="dewp"/>
    </verticalaxis>
    <renderer type="line">
      <option name="linecolor" value="magenta"/>
      <option name="linewidth" value="2"/>
      <option name="missingvalue" value="-9999.9999"/>
      <option name="missingop" value="EQ"/>
    </renderer>
  </plot>
  <data>
    <variables>
      <variable id="time" type="datetime"/>
      <variable id="temp"/>
      <variable id="dewp"/>
    </variables>
    <values>
      20080101000000, 41.1, 24.3
      20080102000000, 23.6, 8.9
      20080103000000, 20.8, 8.7
      20080104000000, 27.3, 12.6
      20080105000000, 36.3, 19.7
      20080106000000, 44.7, 36.8
      20080107000000, 48.3, 41.9
      20080108000000, 54.8, 46.2
      20080109000000, 56.7, 45.6
      20080110000000, 46.7, 42.2
      20080111000000, 52.6, 44.2
      20080112000000, 40.5, 32.8
      20080113000000, 39.6, 33.0
```

Figure 1. MUGL file format

differentiate multiple datasets within the generated graphs. We solved the problem of developing a way to make Dr. Khasanov's data useful by using Multigraph's MUGL file format (Figure 1). In order to work with data submitted from various sources and in various formats, standardization is required. Multigraph also met the requirement of being open source. The software is licensed under the MIT software license agreement. Avoiding a proprietary data format or proprietary methods for rendering graphs of the data ensures that the project is scalable and extensible. Multigraph provided us the ability to standardize the data models and simplify the model portion of CodeIgniter's Model-view-controller process. Since Multigraph provided the formatting for our data, we only then needed to design a database in which the datasets could reside.

3.2 Database Design and Integration with the Framework

The database design was a gradual process, developed and adapted based on our discussions with Dr. Khasanov and our evolving understanding of the intended use of the application. The first step was setting up tables in the database that enable users to create an account and log in. Each user would also have the ability to add personal information to a profile, as well as to edit settings that affect the user's experience of the application. We knew we would have to store potentially large amounts of raw data associated with descriptive information about each dataset, including the material observed (which could be in various formats, from verbal descriptions to molecular formulas), the name and contact information of the person who uploaded it and links to a calibration set, which normalizes the data along the y-axis. Once uploaded, the raw data and the calibration set needed to be converted into the MUGL file format for use with Multigraph, so we would also need to store links to both the MUGL file and the original files. One of the more important functionalities of the application would be the ability to compare and manipulate graphs. Although it was not a priority, we wanted to build in the necessary infrastructure to eventually store certain comparisons and manipulations that the user wants to keep for later reference. By building in the ability for users to save their activities from one work session to the next, we have set up the application to function as more than just a reference; it can essentially operate as the user's "lab notebook".

3.3 User Interface Design and Development

The design and development of the user interface was guided by our intent to encourage users to utilize the application as not just a repository for sharing and retrieving datasets, but also as a platform that would enable various means of interfacing with the data. Initially, the application was envisioned as a simple tool to visualize and manipulate spectroscopy datasets. We quickly realized that our application should aim provide the users with an experience that could enrich their work flow and potentially upgrade it considerably. The data 'viewport' evolved into a 'workspace' that contains organizational components for storing, browsing, notating and visualizing datasets. With the greater scientific community in mind, we brainstormed useful features and organizational components for the application. The interface design was developed with Dr. Khasanov's input in terms of necessary and simple features, such as browsing, uploading, and zooming. The paramount user interface goal was a responsive, clean, clutter-free design that belies the quantity of data underlying the display. According to information visualization guru, Edward Tufte, "[…] confusion and clutter are failures of design, not attributes of information. And so the point is to find design strategies that reveal detail and complexity -- rather than fault the data for an excess or complication.[9]

The decision to use Twitter Bootstrap was made after our first meeting with Dr. Khasanov in which we discussed the basic requirements of the application. Twitter Bootstrap is a simple and flexible HTML, CSS, and JavaScript framework for popular user interface components and interactions.[10] It comes with a base layer of CSS, a fluid or responsive grid system, and JavaScript plugins. There are several frameworks that offer similar tools for rapid prototyping, but Twitter Bootstrap is the most comprehensive. It includes many of the most commonly used user interface components for building websites or web applications, and it has developed a large online community and user base, providing a crucial resource for design and development issues and inquiries. Some other major benefits of Bootstrap include cross browser support, great documentation, and built-in responsive design, which adjusts the layout to fit the screen size of the device being utilized. We felt confident that our user interface development and subsequent user experience would be thoroughly supported by the features of Twitter Bootstrap.

Throughout the design process, we focused on not only ensuring that the application fulfilled all necessary functionality requirements, but also that it would be extensible. One of the primary goals was to build a solid foundation upon which new features, modifications, and customizations can be made to adapt the application for the future needs of not only Dr. Khasanov and his colleagues, but also by other research groups in the scientific community. The complete interface was designed and built, at its most basic level, as a tool for any scientific discipline wanting to upload, visualize, interact with, and organize data files.

3.4 Collaborative Development through Code-Sharing

The model-view-controller process up to this point was not yet integrated to the point of providing the intended functionality. We had developed basic view templates and a scalable data model, but the controllers were the key to enabling the full range of interactivity. We began working more collaboratively in our meetings to build these

controllers, which update the view templates based on the data in our database. However, working in these meetings at every level of the model-view-controller process became cumbersome. The team programming methodology of attacking sub-problems in pairs quickly became outdated and ineffective. Individually, we had been maintaining version control on our own development machines, which began to generate conflicts when two or more of us made changes to the same source code at the same time.

We centralized the version control of our software and solved this problem in our research method by using the collaborative programming website GitHub. GitHub is a source code hosting service that encourages community collaboration.[8] The website uses Git, a version control program for developers which focuses on providing the ability to branch code. Branching in software development, from a conceptual standpoint, means to segment source code that is under revision. From a practical standpoint in our research, using Git's ability to branch source code provided each developer the ability to have local versions that could be worked on independently. After team members made the necessary revisions to their "branched" source code, they merged those branch changes with the central source code. GitHub brings the individual work of the team members together by displaying each branch and merge action in a convenient timeline.

Furthermore, GitHub fulfills Dr. Khasanov's goal of allowing the project to be scalable and extensible. This is because GitHub offers free source code hosting for open source projects. The project may scale in the sense that all of its source code is persistently hosted by GitHub and accessible on the Internet. Extensibility is offered by using GitHub because the website uniquely offers the ability for the global programming community to view, change, download and branch the source code for the project, once the initial development team makes the project public. Changes to the models and controllers can be made by anyone with an interest in the project because we chose GitHub as a solution to the slower, more iterative team-programming process we had started with. Software users can modify the design of the view templates. Only by using GitHub were we able to allow scientists the ability to perform their own localizations, customizations, and design changes to the flexible view templates that we created using Twitter Bootstrap. In this way, GitHub fulfilled another one of Dr. Khasanov's goals by allowing the project to be easily localized for scientists outside the United States.

4. Results and Conclusions

In the first three months of the project, we have successfully developed a functional proof-of-concept (see Figure 2.) for a web application capable of storing, standardizing, visualizing, organizing and sharing multivariate spectroscopy data uploaded in a range of different formats. By using the robust model-view-controller infrastructure, existing open source libraries, and GitHub's collaborative code-sharing development platform, we have also created a highly adaptable, extensible application that encourages others teams of scientists and programmers to utilize and further develop according to their specific fields and needs. The application is dependent only on robust open source projects that are stable and implemented as industry standards. The server we used to host the web application ran Apache web server, MySQL, and PHP5. We also installed the PHP5 modules for MySQL. Finally, we installed Python to run the scripts necessary to process uploaded datasets into MUGL files. Python's extensive data processing modules could also be used later for calibration calculations. Any system capable of running these common software programs should be capable of running the application.

Dr. Khasanov was somewhat critical of our implementation when it came to some extra features we felt were important. For example, we implemented a note-taking feature into the application, intended as a means for users to record information about their progress or about particular datasets. He felt that many scientists would simply not need this feature. However, although Dr. Khasanov and his colleagues will be the primary test users, in order to meet our mutual goal of establishing an application that can eventually be utilized by a global audience, we must take into account and build features that will likely be useful for others, based on our knowledge of existing data analysis tools. Dr. Khasanov did not have much feedback on the visual design of our application. He expressed trust in our skills and ability to design the application's visual features like the color scheme, layout and interface structure. The most feedback and contention was around the interactivity of Multigraph, which Dr. Khasanov was acutely concerned about. His concerns were primarily over the adequacy of Multigraph's default functionality to scale, pan and zoom on graphs. After much discussion and exploration of Multigraph's capabilities, we concluded that it would serve the immediate purpose of providing the desired graph functionality regardless of any perceived inadequacies, and if needed, Multigraph could be replaced by more customized graphing tools at a later point in time.

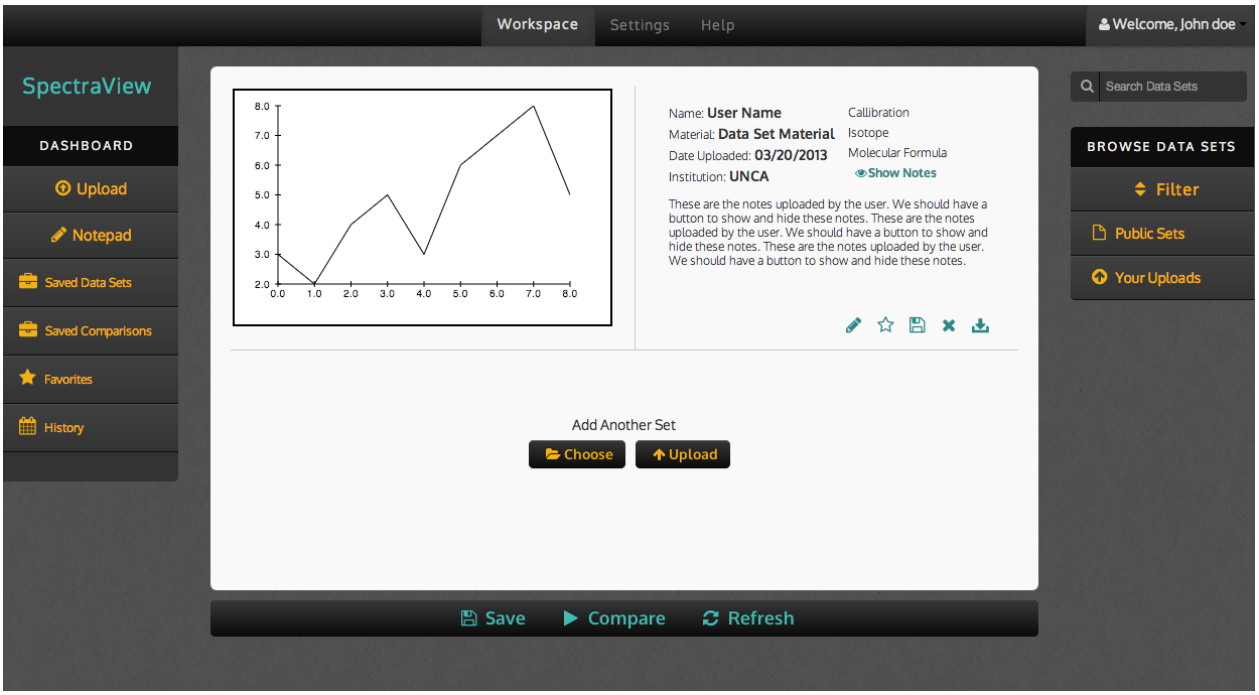

Figure 2. Screen capture of the functional prototype

Our future plans for the project involve removing the dependency on Python for the conversion and creation of MUGL files. We would also like to provide the ability for users to compare datasets within the same graph. We have discovered that Multigraph is capable of this functionality, after some customization. Furthermore, we would like to implement Dr. Khasanov's goal to calibrate datasets automatically. This will require converting his algorithms to software code, which can then be run on the server.

The project will remain open source[11] for these modifications to be made by any individuals who desire to work on the application. Since all the software we used for the application is open source, programmers may continue to build upon, adapt and customize the application for a variety of purposes. This is also advantageous for the community of scientists involved in the study of spectroscopy, as the software can be run anywhere without having to be concerned about proprietary licenses. The software can be localized and adapted for any spectroscopy lab, or it can be used, as intended, for collaboration in the replacement of relatively outdated dataset visualizers and comparison tools. We are confident that this development process and the tools we have utilized create the foundations of a promising, effective model for future collaborations between the scientific community, programmers and designers. Much of the software presently used could be effectively reworked to adapt to new and flexible web technologies that have robust, built-in support for managing, interpreting and sharing data.

5. Acknowledgements

We are grateful for Dr. Airat Khasanov's subject matter expertise, as well as his input and encouragement, without which this project would not be possible. Dr. Semmy Purewal's recommendations to use MultiGraph and GitHub were central to the success of the project.

6. References

1. NASA, "Introduction to Spectroscopy", http://imagine.gsfc.nasa.gov/docs/teachers/lessons/xray_spectra/background-spectroscopy.html.
2. Mössbauer Effect, http://www.mossbauer.org/history.html

3. Daniel Rodionov, Data Archive at Mer Moessbauer Spectrometer, http://iacgu32.chemie.uni-mainz.de/mer/.
4. M. Darby Dyar, Mars Mineral Spectroscopy Database, https://www.mtholyoke.edu/courses/mdyar/marsmins/.
5. "Spectral Database for Organic Compounds SDBS", http://sdbs.riodb.aist.go.jp/sdbs/cgi-bin/direct_frame_top.cgi.
6. "Code Ignighter User Guide 2.1.3", http://ellislab.com/codeigniter/user-guide/.
7. RENCI, "Multigraph: Interactive Data Graphs for the Web", http://multigraph.GitHub.com/docs.
8. Jennifer Marlow and Laura Dabbish, "Activity Traces and Signals in Software Developer Recruitment and Hiring," In *Proceedings of the 2013 conference on Computer supported cooperative work* (CSCW '13). ACM, New York, NY, USA, 145-156. DOI=10.1145/2441776.2441794 http://doi.acm.org/10.1145/2441776.2441794.
9. Edward Tufte, Envisioning Information, Cheshire CT (Graphics Press, 1990).
10. "Twitter Bootstrap: Scaffolding", http://twitter.GitHub.com/bootstrap/scaffolding.html.
11. OpenSpectra open-source GitHub repository, http://GitHub.com/ntbnnt/openSpectra.