Transforming Introductory Computer Science Projects via Real-Time Web Data

Austin Cory Bart, Eli Tilevich, Simin Hall, Tony Allevato, Clifford A. Shaffer
{acbart,tilevich,thall57,allevato,shaffer}@vt.edu
Virginia Tech

ABSTRACT

While computing is becoming increasingly distributed, programming projects in introductory classes remain mostly divorced from the student's day-to-day computing experiences. These experiences entail interacting with real-time Web-based data from sources that include weather reports, news updates, and restaurant recommendations. The disconnect between student experiences and the content of their programming projects is known to drive some students away from computing. In addition, to adequately prepare students for the realities of modern software engineering, educators should introduce issues pertaining to distributed computing early in the curriculum. To address these problems, we have created RealTimeWeb—an architectural framework that makes real-time web data accessible for introductory programming projects. The framework effectively introduces important real-time distributed computing concepts without overwhelming students with the low-level details that working with such data typically requires. Preliminary results indicate that our approach can be effective in the context of a typical CS2 course, and that real-time data is relevant to students. RealTimeWeb libraries and associated resources are publicly available for use, with multiple language bindings to many real-time data sources. A rapid-prototyping tool available through the project's website facilitates the development of client libraries with easily accessible APIs for new real-time Web-based data sources.

Categories and Subject Descriptors

K.3.2 [**Computer and Information Science Education**]: Computer Science Education

General Terms

Design, Human Factors, Reliability, Experimentation

Keywords

distributed computing; projects; introductory courses

SIGCSE'14, March 5–8, 2014, Atlanta, GA, USA.

ACM 978-1-4503-2605-6/14/03 ...$15.00.
http://dx.doi.org/10.1145/2538862.2538941.

1. INTRODUCTION

Most people's experience with computing today involves distributed Web architectures and continuously updated, remote data sources. College students consume real-time Web-based data on their mobile devices. Examples abound. Weather information, traffic data, stock values, shopping deals—all of these real-time data sources are available as remote Web-based services that are then integrated into web-based applications available through browsers or mobile apps. Interacting with Web-based real-time data influences students' perception of what Computer Science is. This perception comes with them when they embark on studying the discipline.

Computing educators might do well to introduce issues related to real-time web data into the curriculum as early as possible for two reasons. First, meeting students' expectations that Computer Science studies exciting topics that are relevant to the students' experiences as computer users can help attract and retain good students [2, 1, 7]. Second, since distributed real-time data has entered the mainstream of computing, students should be introduced to the corresponding technical issues to prepare for the modern IT workforce. Indeed, the new ACM/IEEE Computer Science Curriculum 2013 advises 10 hours of material on a new dedicated *Networking and Communication* Knowledge Area [9].

Introducing real-time data is not only about increasing engagement. Perhaps more important, it is an avenue for exploring computing in a social context. The CS Curriculum 2013 emphasizes the importance of Social Issues [9], prescribing 16 hours on topics such as Social Context and Data Privacy. The ability to access real-world data streams enables introductory learners to deal with real-world problems. Consider assigning students to explore how geological data about earthquakes can be used to aid disaster relief. Another project might have students mine political data to find evidence of corruption. Yet another project, perhaps controversially, would involve analyzing social media data provided by services including Twitter or Facebook, to identify the cases of private data being left publicly available unintentionally. Tying actual class projects to such topics can powerfully convey computing's role in society to the introductory learner.

Alas, despite its ubiquity, incorporating distributed computing into the curriculum has proved difficult due to lack of educational resources. Even the seasoned software developer finds working with network protocols and parsing binary data streams conceptually complex, particularly in the presence of overwhelming technical issues that range from

handling partial failure to dealing with distributed components evolving independently from each other. Not surprisingly, a common CS curricular design strategy is to leave the challenges of teaching distributed computing for later courses, when students will have accumulated sufficient technical expertise and tolerance for engineering non-trivial system designs.

The net effect of these curricular design choices is that the content of introductory courses remains isolated from issues pertaining to Web-based real-time data. Programming projects are particularly vulnerable to this omission. When working on programming projects, introductory students find themselves disconnected from familiar data from social media, news outlets, and local business. Instead, they find themselves given abstract, toy problems with limited context in their lives. Divorcing the content of programming projects from the students' experiences as computer users has a debilitating effect on motivation and engagement [2]. In fact, competent students are known to leave the major, having been discouraged by the lack of relevance for what they learn [1]. This problem is particularly acute for female students, for whom real-world application has been identified as the primary motivation for studying the discipline [7]. Our vision is that introductory courses can overcome the technical barriers inherent in working with real-time web data, enabling us to introduce novel, relevant projects to students.

2. PRIOR WORK

Insulating beginner programmers from complicated systems while still providing the ability to manipulate interesting data is not a novel concept. For example, most programming environments designed for novices feature convenient methods for manipulating images and sounds. Racket [6] treats images as a primitive data type, avoiding the complex, low-level programming that is typically required. While images and sounds have an obvious appeal, they lack the direct real-world connection afforded by real-time data. However, little work has focused on creating similar student-oriented interfaces for real-time data sources.

An example of a positive impact of working with real-time data on student learning is a project at Stanford in a Geological Sciences course. In this project, students worked with real-time data on earthquakes [5]. Although students did not program, scientific computational tools were used to analyze the data and reach conclusions. The instructor reported that students became significantly more engaged as they worked with data that had relevance to them. For instance, many students became excited when they discovered that there were earthquakes happening in their region all the time. The final assignment had students choose a city that they wanted eventually to live in and determine the geological risks of the area. The instructor reported that, even after the course had ended, students applied similar analysis to other geographical regions relevant to their lives. This assignment contextualized the learning experience for the student, and created "a personal connection and positive affect that motivates their future learning" [5]. Corresponding projects, perhaps even using the same data source, are ripe for a programming class.

Similar projects have been used in statistics courses [3] and data mining courses [10]. Although other domains seem ready and willing to bring real-world data into the classroom, the research literature on the topic is scarce. A project conducted by Dr. Marc Waldman introduces realistic open data into upper-level database courses [11]. In essence, this project challenges experienced learners by taking advantage of the complicated nature of real-world data. While our primary design objective is to accommodate novice users, we also strive to appropriately challenge more advanced users, enabling them to develop higher-level skills.

3. OUR APPROACH

We have created and distributed a software architecture framework (named "RealTimeWeb") that provides introductory programming students with an easy way to manipulate distributed real-time data. Our approach offers technical scaffolding for the students to gradually ease into (or completely circumvent if appropriate) some of the most vexing complexities of distributed computing. At the heart of our project are carefully engineered client libraries through which students can access the data provided by real-time web services. These libraries are readily available through an online curated gallery, designed to be quickly adapted to instructors' specific academic needs. This gallery also provides a tool for rapidly prototyping new libraries based on our framework.

3.1 Client Libraries

To connect students to real-time data sources, we have designed client libraries with features intended specifically for novice programmers. Each library offers a selection of method calls to request, parse, and return real-time data. Presently, we have created libraries to provide

- Business reviews from Yelp.
- Weather forecasts from the National Weather Service.
- Stock trading information from Yahoo Finance.
- Global earthquake reports from the US Geological Survey.
- Content from link-sharing site Reddit.com.

In theory, any publicly available online real-time data source can be targeted by our framework, and we intend to continuously offer new and improved client libraries.

3.1.1 Language

There are many different programming languages used in introductory CS courses [9]. To account for this, we implemented each library in a number of common beginner languages, including Python, Java, and Racket. We have also made an effort to provide compatibility on key platforms, including Android.

3.1.2 Scaffolding

The instructor can determine how data is returned, so as to provide varying levels of scaffolding. Choices include raw strings of data, semi-structured hashes and lists, or object-oriented classes. Some libraries are available in reduced variants that return atomic data (e.g., a derivative of the Weather Service client library returns only the current temperature, as opposed to a complete forecast including humidity, wind, etc.). In this way, how much work students must perform to manipulate the data can be altered to suit the class. This means that the libraries can be used in a wide variety of educational scenarios.

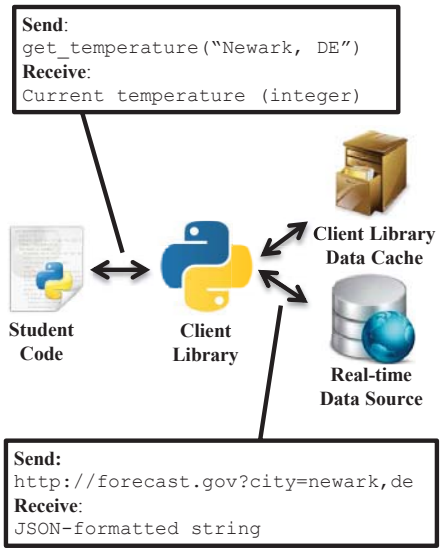

Figure 1: Client Library Architecture

3.1.3 Internalized Data Cache

Perhaps the single most useful feature of the libraries is the *internalized data cache*. The cache can be used to avoid making requests directly to the actual data source, instead accessing a local, static data store (a file). Figure 1 demonstrates the flow of data when using a client library. The cache option offers a number of advantages.

Idempotency By its nature, real-time data is subject to rapid temporal changes. Weather forecasts change on a daily basis, for example, and services like Reddit or Twitter change by the minute. Developing a program that uses such volatile data can be tricky, since it can change between runs of the program. To accommodate introductory students, the libraries make it possible to used cached local data, so consistency of input from run to run is guaranteed.

Consistency The content and structure of the web is highly dynamic, as are the web services composing it. Because service APIs commonly evolve at a dizzying pace, the libraries that depend on them must be updated accordingly. Even during a single semester, changes in the data source could be introduced that would result in an out-of-date library. However, this problem can be bypassed by keeping to the internal data cache until an updating fix is released, avoiding any serious delays in development.

Connectivity Although most campuses have gigabit internet connections, this is not universal. Additionally, many students live in off-campus settings with varying internet capabilities. Requiring students to develop an internet-based application with poor internet connections can be a tall order. However, the the cache allows client libraries to be used offline, if necessary.

Efficiency Even assuming a fast and stable connection, the performance of many web services is limited by the number of connections that the server can handle. Student developers can be polite consumers by developing with client-side data, greatly reducing the number of calls to the online service. This is especially important since many APIs throttle the number of requests (e.g., 100 API calls per day), sometimes penalizing abusers with time-outs and bans. To avoid this, web service clients typically use a polling architecture (with an artificial delay between requests); running with local-data negates the need for the delay, and makes large-scale testing practical.

Tests Instructors can use the cached data for testing, ensuring uniform coverage by all students. There is no need to worry about students missing out on edge cases not covered during a particular time interval, since they can be provided with the important cases in test files. Since the cache is stored in a convenient JSON-based data format, it can easily be modified by instructors to return specific results. The ability to ensure consistent tests is particularly useful for automated grading systems, such as Web-CAT [4].

3.1.4 Threading

To address the needs arising in an assortment of situations, the libraries offer the flexibility of returning the results of API calls either synchronously or asynchronously. Concurrent programming, although an important skill for students to develop, can be overwhelming for beginners. Even as work to integrate multiprogramming into lower-level courses moves forward, scaffolding is still required if we want intro students to develop applications with parallel processing capabilities. Using the asynchronous return mode, the hard problems of threading can be avoided.

3.1.5 Open-source

The libraries are meant to be good examples of API design. As students gain mastery, they can be encouraged to read the source code to learn how network communication and data parsing is implemented. They can then modify, extend, and even re-implement the API as they see fit. If students choose to create their own API based on our model, they can submit it to our gallery and gain wider recognition for their work.

3.1.6 Examples

Figure 2 demonstrates using the Java version of the Earthquake library. This sample program polls the online service for new earthquakes and then notifies the user. This polling architecture is a common pattern for web-service consumers, so it is useful to expose it to students. Note that if line 12 had been omitted, the library would have returned results from its internal cache instead of the online service. In turn, this would allow us to remove the `Thread.sleep` in line 32, and receive huge speed boosts. Also note the use of a `HashSet` - since data is returned from the past hour, but accessed every five minutes, there is a significant amount of redundant data that must be filtered out. The local cache replicates this redundancy for complete authenticity.

Figure 3 shows an example of using the reduced Racket WeatherService library, which only exposes a method to get the current temperature of a city.

3.2 Curated Gallery

Complete libraries and documentation are available at `http://research.cs.vt.edu/vtspaces/realtimeweb/`. Besides the various language bindings available for a service

Figure 2: A simple program demonstrating the Java Earthquake library

```
import java.util.List;
import java.util.HashSet;
import realtimeweb.earthquakeservice.main.EarthquakeService;
import realtimeweb.earthquakeservice.domain.Earthquake;

public class EarthquakeDemo {

    public static void main(String[] args) throws EarthquakeException {
        // Use the EarthquakeService library
        EarthquakeService es = EarthquakeService.getInstance();

        es.connect();  // Remove to use the local cache

        // 5 minute delay, but if we use the cache no delay is needed!
        int DELAY = 5 * 60 * 1000;

        HashSet<Earthquake> seenQuakes = new HashSet<Earthquake>();

        // Poll service regularly
        while ( true ) {
            // Get all earthquakes in the past hour
            List<Earthquake> latest = es.getEarthquakes(History.ALL);
            // Check if this is a new earthquake
            for (Earthquake e : latest) {
                if (!seenQuakes.contains(e)) {
                    // Report new earthquakes
                    System.out.println("New quake!");
                    seenQuakes.add(e);
                }
            }
            // Delay to avoid spamming the weather service
            Thread.sleep(DELAY);
        }
    }
}
```

Figure 3: A simple program demonstrating the Racket Weather library

```
(require "weather-offline.rkt")

; string -> string
; Consumes a city and returns whether its
; current temperature is "hot" or "cold"
(define (report-weather city)
    (cond [(< 70 (get-temperature city))
           "hot"]
          [else "cold"]))

; We use the offline version of the library
; for consistent check-expects
(check-expect (report-weather "Nome, AK")
              "cold")
(check-expect (report-weather "Miami, FL")
              "hot")
```

(e.g. Python, Racket, Java), there are a number of other useful pieces of information:

- API documentation and student-oriented user guides for each language and library.
- Alternative datasets for the internalized data cache (e.g., instead of business reviews from around "Blacksburg, VA", there might be another dataset for "Indianapolis, IN").
- Reduced variants of the libraries for targetted assignments.
- Example assignments that use the library.

All resources are open-source and fully supported. They are being continuously refined and extended.

3.3 Prototyping Tool

An important byproduct of our project is the creation of an online tool for rapidly prototyping new libraries. Most of the code used in our libraries follows the same pattern for any given language. First, requests are made to a web service and raw data is returned (typically as XML or JSON). Next, the data is parsed into some intermediary, semi-structured form using dictionary and list types that are native

to the language. Finally, the data is encoded into a read-only class or struct, depending on the disposition of the language. For example, beginner students using Racket might deal with structs, instead of classes. This is not true in object-oriented languages such as Python and Java.

Because the data flow is consistent regardless of programming language or data source, we can leverage this similarity to fill out a template for the target languages based on a single, abstract meta-description (a "Client Library Specification"). The JSON-formatted Specificiation contains two parts: information about the methods exposed by the web service and the structure of the data returned. The system is compatible with several methods of sending arguments to web services, including Query Strings, POST Variables, and URL-Embedded data. Once a user has created a Specification, the tool generates valid and functionally-equivalent Racket, Python, and Java code to access the web services. These client libraries come ready-made with documentation provided through the Specification, and their source code is extremely straightforward to comprehend and extend.

The tool has already been successfully used to create five of the client libraries. The automatically-generated code usually requires only a few minor, manual tweaks before it is ready for public exposure. In fact, the most recently-created library (for connecting to a Magic: The Gathering card database) took less than an hour to specify, generate, and refine. The tool, documentation, and examples are available online as a component of the gallery.

4. TRANSFORMING COURSE PROJECTS

When integrated into a class project, our tools offer several features that positively impact student engagement. First, the availability of libraries for a multitude of services creates options for free-form projects, in which students enjoy greater *autonomy* in development. Second, the abundance of our library offerings also increases the chances of students finding data sources that are truly *relevant* to their interests. Third, the simple design of the libraries fosters a sense of *self-efficacy* and *competence* within the student; they can work with complicated real-world data without dealing with the frustrating real-world complications. These aspects are all recognized as increasing intrinsic motivation and engagement [8].

However, using real-world data does more than just engage students and increase their intrinsic motivation. Instructors can use the libraries to introduce real-world problems. Example project ideas include:

- Reddit Service: Social link-sharing website Reddit aggregates interesting content from around the web, including news and other real-time data sources. Students could be tasked with processing data from the site and finding interesting patterns in comments and posts. Even with only a rudimentary knowledge of string parsing, there are many interesting operations that can be performed, such as analyzing average comment length or finding a keyword.
- Weather Service: Planning a long-distance trip can be tricky, and preparing for weather is important. Students could investigate how to avoid bad weather between cities using different search algorithms to find best possible paths.
- Earthquake Service: Earthquakes happen around the world, all the time. Students could use a spatial data structure such as a Quadtree to record and analyze data about earthquakes and how they affect the world.

As a specific example of the potential educational benefits that RealTimeWeb can afford, consider two alternate sketches for a programming project that can be assigned to reinforce the topic of circular linked lists, a fixture of a typical data structures course. The first version uses abstract data, while the second one uses the data provided by the Business Service library that comes as part of the current RealTimeWeb catalog.

Original Version:

1. Create a circular list data structure capable of holding heterogeneous data.
2. Create a Data class that contains an integer ID and a String description.
3. Place Data objects whose ID fields represent consecutive integer values into each node of the list.
4. Remove all the nodes containing Data objects, whose ID fields are even numbers.

RealTimeWeb Version:

1. Create a circular list data structure capable of holding heterogeneous data.
2. Download a list of nearby restaurants and place them into the nodes of the list, sorted by their price levels.
3. Remove all the restaurants that do not offer vegetarian options.

Although both versions reinforce the same topic of circular linked lists and require a comparable amount of student effort, the RealTimeWeb version has students manipulate data they are likely to encounter in their day-to-day experiences, thus potentially increasing motivation and engagement.

It is important to recognize that integrating these services needs to be more than skin deep. Students should be given a situated perspective on how these data streams interact with the real world. Rather than just downloading an abstract list of data, assignments should lead to practical, interesting tools and results that the students can be proud of.

5. FORMATIVE EVALUATION

5.1 Usability Survey

We piloted our Business Service client library in a CS2-style course offered at Virginia Tech (CS 2114) during Spring 2013. This course covers topics typical to a second semester Computer Science course, including Object-Oriented concepts and the Java programming language. All students in the class are Computer Science majors and minors in their first or second year, with mostly limited prior programming experience. Every week, students use paired programming to complete a lab assignment. There are also three large projects meant to be completed over the course of the semester.

During one lab session, students were assigned a multi-part problem that required the use of the Business Service library. To personalize their experience with the library, the internal data cache was filled with data from the Blacksburg area. The description of the lab began with an overview

of the problems and advantages inherent in using real-time data, and then described how the Business Service library could be used. In the first task, students used the library to create a list of highly rated businesses in the area. In the second, they expanded that list with more detailed information. The primary intent of this lab was to familiarize students with the client library.

Students had a second opportunity to work with the library in the third project of the course, where they built a more complicated "Restaurant Guide" Android application. Students again searched for a list of restaurants in a given region. This time, however, they were required to place the results in their implementation for a circular linked list data structure.

A voluntary survey was administered to the class after students had completed the project, and 17 responses were gathered. Overall, the results were positive. All but one student indicated that they found the API "applicable to day-to-day computing experiences", and all students thought that the API was "easy-to-use". Qualitative feedback from the students highlighted "clear, easy-to-figure-out methods" and that they "provided information that was easy to work with".

5.2 Relevancy Survey

In addition to this formative evaluation, a large-scale survey was conducted of 370 undergraduates at Virginia Tech in the Fall 2013. 68% of the respondents were Sophomores and Juniors, 87% were male, and 68% were Computer Science majors. Students were asked to gauge how much learning a given technology would benefit their future careers. Working with Real-time Data compared favorably with Mobile App Development and Website/Web Application Development (with most students feeling they would benefit or definitely benefit from these technologies), as opposed to Programming Robots, Game Development, and Raspberry Pi Development (with most students feeling somewhere between neutral and beneficial). This is extremely encouraging, especially since our RealTimeWeb framework is designed to be compatible with Mobile technologies and back-end web application development.

6. FUTURE WORK

Currently, there are five services available through our client libraries, each in three languages. Ultimately, we plan to have an extensive gallery that can cater to a wide range of student interests. We are particularly interested in soliciting feedback from the community on other services that would be of interest and languages that should be supported.

Although the results from our formative evaluation were positive, it is important not to overstate them. More surveys and careful evaluation will be required to determine the exact impact that real-time data assignments offer educators. We are using RealTimeWeb in programming projects during Fall 2013 at Virginia Tech in a sophomore/junior-level data structures course, and at University of Deleware in a freshman-level course.

7. CONCLUSION

Learning to handle real-time web data is an important skill for modern CS students to develop. This learning experience also makes introductory programming projects more relevant to the students' day-to-day computing experiences, thereby increasing engagement and motivation. In order to reduce the technical barriers inherent in interacting with distributed systems, we have created publicly available client libraries that make it possible to connect to popular web services in a student-friendly manner, enabling a gradual introduction to the complexities of distributed computing. Results from an early pilot indicate that our libraries are easy to use by introductory students. The RealTimeWeb framework provides a promising way to better engage students while preparing them for the realities of the modern IT workplace.

Acknowledgments

This research is supported by the National Science Foundation through the Grant TUES #1140318.

8. REFERENCES

[1] L. Carter. Why students with an apparent aptitude for computer science don't choose to major in computer science. In *Proceedings of the 37th SIGCSE technical symposium on Computer science education*, SIGCSE '06, pages 27–31, New York, NY, USA, 2006. ACM.

[2] D. I. Cordova and M. R. Lepper. Intrinsic motivation and the process of learning: Beneficial effects of contextualization, personalization, and choice. *Journal of educational psychology*, 88:715–730, 1996.

[3] A. B. Downey. *Think Stats*. O'Reilly Media, July 2011.

[4] S. Edwards. Using test-driven development in the classroom: Providing students with automatic, concrete feedback on performance. In *In Proceedings of the International Conference on Education and Information Systems: Technologies and Applications*, EISTA, 2003.

[5] A. E. Egger. Engaging students in earthquakes via real-time data and decisions. *Science*, 336(6089):1654–1655, 2012.

[6] M. Felleisen, R. B. Findler, M. Flatt, and S. Krishnamurthi. *How to Design Programs*. Second edition, 2012.

[7] A. Fisher, J. Margolis, and F. Miller. Undergraduate women in computer science: experience, motivation and culture. In *Proceedings of the twenty-eighth SIGCSE technical symposium on Computer science education*, SIGCSE '97, pages 106–110, New York, NY, USA, 1997. ACM.

[8] R. M. Ryan and E. L. Deci. Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. *The American psychologist*, 55(1):68–78, Jan. 2000.

[9] M. Sahami, S. Roach, E. Cuadros-Vargas, and D. Reed. Computer science curriculum 2013: reviewing the strawman report from the acm/ieee-cs task force. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, SIGCSE '12, pages 3–4, New York, NY, USA, 2012. ACM.

[10] L. Torgo. *Data Mining with R, learning with case studies*. Chapman and Hall/CRC, 2010.

[11] M. Waldman. Keeping it real: utilizing nyc open data in an introduction to database systems course. *J. Comput. Sci. Coll.*, 28(6):156–161, June 2013.