CloudDB: Components for Exploring Shared Data with MIT App Inventor

Natalie Lao
Department of Electrical Engineering
and Computer Science
Massachusetts Institute of Technology
Cambridge, Massachusetts
natalie@mit.edu

***Abstract*—Data sharing through Cloud technology is one of the most powerful new computer science concepts of the past few decades. As such, developing powerful and easy-to-use tools for incremental learning and application of shared data concepts is an important endeavor. My work focuses on using MIT App Inventor, a popular blocks-based mobile application development tool for teaching computational thinking to young students, to make shared data technology understandable and usable by anyone without the need for extensive computer science training. I present the ongoing development of CloudDB, a set of coding blocks for MIT App Inventor that allows users to store, retrieve, and share various types of data in tag-value pairs on a Redis server for their mobile applications.**

Keywords*—*blocks, cloud, data sharing, App Inventor, mobile application development, computational thinking, education.

I. Introduction

Shared data is a new concept that has vastly changed the technological landscape over the past decade. It has led to innovations such as Dropbox, Google Docs, and essentially every modern scalable web service. Educating young students on its applications will enable them to understand and contribute to data sharing technology quickly, easily, and adeptly. I present (1) an approach for creating a technical system with simplicity and usability as its design goals that allows kids to interact with shared data within reasonable abstraction, (2) a detailed implementation of the working CloudDB tool, which was created using the Redis database, cache and message broker, and (3) plans for further development and release of CloudDB to the public through MIT App Inventor.

II. Background

A. Cloud capabilities of MIT App Inventor

Due to its low barrier of entry, MIT App Inventor is being used as the primary teaching tool for many introductory programming classes and workshops targeting students from late elementary school to professional level [1]. Its accessibility to young students and popularity as an international computer science teaching tool makes it a great platform for distribution of and research on shared data programming tools. MIT App Inventor has FirebaseDB, an experimental component that developers use to share data across multiple devices [2]. However, the component relies on a cloud data store service provided by Google, which will soon be tightly integrated with the Google Play Store, in effect requiring developers who wish to use the Firebase database to register their applications on the Play Store. This is incompatible with much of MIT App Inventor's education-focused use cases, so this method of providing cloud storage capabilities for MIT App Inventor is quickly reaching its end-of-life.

The CloudDB component is scheduled to replace FirebaseDB, and is heavily modeled after its predecessor. It is important that current FirebaseDB users on MIT App Inventor are able to quickly and easily transition their FirebaseDB projects over to CloudDB and to continue creating projects that use shared data without much of a learning curve. A similar set of blocks will best help achieve this goal.

B. Data sharing tools in Scratch

There are two notable projects for shared data education for primary school students in the Scratch programming language: Shariables and Cloud data-structures. In 2007, Stern implemented Shariables for Scratch, which were variables stored on a server that could be modified across multiple projects and multiple users [3]. These Shariables had server-side persistence and had basic access control features. In 2013, Dasgupta implemented Cloud data-structures for Scratch 2.0, which extended scalar variables and lists in Scratch through a boolean property, thereby enabling Scratch programmers to store and retrieve data through their projects [4]. Cloud data-structures were both persistent across multiple execution instances and shared between simultaneous instances. As a blocks-based programming environment, Scratch's coding interface is similar to that of MIT App Inventor. However, Scratch projects are meant to be played and viewed on the Scratch website whereas MIT App Inventor projects are meant to be downloaded as apps for mobile devices.

III. Design

I imposed a key requirement in designing the CloudDB component: Since one of the goals of the MIT App Inventor project is to democratize mobile application creation technology, CloudDB must be understandable and usable by young students and generally by anyone without extensive computer science training. Additionally, since CloudDB was envisioned as a replacement for FirebaseDB, I also wanted users of the FirebaseDB component in MIT App Inventor to be able to easily transition to using CloudDB. Thus, I reused many of the same block names from FirebaseDB and followed the prior naming convention for any new blocks that I created.

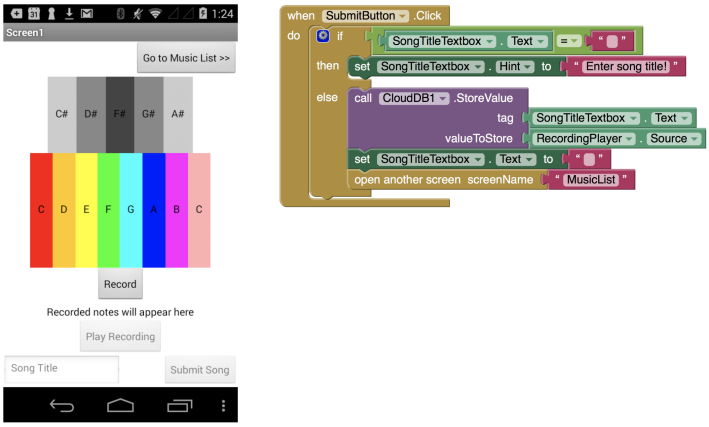

Fig. 1. MusicShare's first screen, which provides a user interface for creating, recording, and submitting music to the database.

Fig. 2. MusicShare's second screen, which lists all of the songs that have been submitted to the repository. The list is updated live. Tapping on a song automatically plays it.

With these considerations in mind, I designed the CloudDB component to include the following features: (1) data buckets identified by just two text properties, *AccountName* and *ProjectID*, (2) a simple key-value pair data storage, modification, and retrieval scheme that works with a text-based key and a value of any type, (3) an event listener that returns the tag and new value any time data in the specific CloudDB project bucket is changed, (4) atomic pop and append list operations, and (5) under-the-hood SSL encryption. This component includes two properties in the MIT App Inventor Designer and fourteen blocks in the Blocks editor.

IV. Implementation

The database was written using the Redis application programming interface (API). Redis was chosen because it is a large and well-maintained open-source project like MIT App Inventor with an expansive selection of features, including easy-to-use Pub/Sub capabilities [5]. It is also an in-memory data storage, which makes it exceptionally fast. Specifically, Jedis, a Redis Java driver, was used for compatibility with the MIT App Inventor codebase.

When a client, such as a phone running a MIT App Inventor application that uses CloudDB, wishes to make a request to view or change data on CloudDB, it first communicates with the proxy server through the Transport Layer Security (TLS) protocol. Once the proxy server validates the request, it sends the request through the Transmission Control Protocol (TCP) to the Redis server. For security purposes, the Redis database only accepts communications from localhost, which in this case is just the proxy server. The Redis server responds to the proxy server's request through TCP. Finally, the proxy server sends the response back to the client through TLS. Client-side programming for the CloudDB component was done in Java and Lua script, Redis configuration directives were used to set up the server, and the Secure Sockets Layer (SSL) proxy server was written in Elixir/Erlang to encrypt CloudDB data.

V. MusicShare: An Example CloudDB App

CloudDB is able to interface with various MIT App Inventor components to create complex apps that allow data sharing. One example of an app that used to be impossible to create in MIT App Inventor without private extensions or significant external programming is the MusicShare app. This app extends a basic piano app and allows users to play, record, and submit their own songs (Fig. 1), as well as listen to songs that others users of the app have shared (Fig. 2). As an upgrade to FirebaseDB and currently functionality of MIT App Inventor, CloudDB is able to automatically recognize and parse media files across different devices.

VI. Future Plans

Several app tutorials and a curriculum plan focused on shared data have been created and piloted for CloudDB. Our team is currently working on improving security, providing access restrictions for project data, and developing log viewing capabilities. CloudDB is estimated to be released on the MIT App Inventor interface in late 2017 or early 2018.

Acknowledgment

Special thanks to Hal Abelson, Jeff Schiller, Evan Patton, Andrew McKinney, Michael Tissenbaum, Josh Sheldon, Karen Lang, Marisol Diaz, and other members of the MIT App Inventor team for all of their knowledge, insight, and support.

References

[1] MIT App Inventor. Massachusetts Institute of Technology, 2015. Web. 03 Aug. 2016. http://appinventor.mit.edu/explore/.

[2] MIT App Inventor. "Experimental Components - App Inventor for Android." MIT App Inventor. Massachusetts Institute of Technology, n.d. Web. 03 Aug. 2016. http://ai2.appinventor.mit.edu/reference/components/experimental.html #FirebaseDB.

[3] T. Stern. NetScratch: A networked programming environment for children. Masters thesis, Massachusetts Institute of Technology, 2007.

[4] Sayamindu Dasgupta. 2013. From Surveys to Collaborative Art: Enabling Children to Program with Online Data. In Proceedings of the 12th International Conference on Interaction Design and Children (IDC '13). ACM, New York, NY, USA, 28-35. DOI: http://dx.doi.org/10.1145/2485760.2485784

[5] "Redis." Redis.io. RedisLabs, n.d. Web. 10 May 2017. https://redis.io/.