ODK Tables: Case Studies in Deployment

Samuel Sudar, Saloni Parikh, Mitchell Sundt, Gaetano Borriello
Department of Computer Science and Engineering
University of Washington
Box 352350
Seattle, WA 98195
{sudars, ssparikh, msundt, gaetano}@cs.washington.edu

ABSTRACT

Open Data Kit (ODK) Tables is an Android application that allows users to enter and curate tabular data. Users can view the data through built-in or custom views using HTML/JavaScript. In this paper we discuss our lessons learned from real-world deployments of Tables. Specifically, our initial aim of providing a customized view of a table did not have a sufficient level of customizability. Rather than simply the views of the data tables, the entire application needed to be navigable using web-based tools that do not require recompilation to make presentation changes. We also discuss issues encountered with making the application scalable when faced with real-world datasets.

Keywords

Open Data Kit, ODK Tables, mobile phones, mobile database, data tables.

1. INTRODUCTION

Data collection on smartphones has become an important component of many ICTD applications. Tools like Open Data Kit (ODK) [1, 2] simplify the act of data collection through the use of forms and increase the quality of collected data. With increasingly powerful mobile tools available, organizations desire more sophisticated data collection capabilities that go beyond simply replacing paper forms with a convenient user interface on a smartphone. ODK Tables [3] is a tool designed to fill this role.

ODK Tables acts as a data manager and viewer. It occupies a middle ground between two common models of storing data in the developing world—spreadsheets (e.g. Excel) and relational databases (e.g. Access). These tools are popular because of their relative ease of use compared to fully-featured database management systems, and both provide valuable functionality: Excel serves to quickly visualize and edit data, while Access supports sophisticated queries and linking data between tables. ODK Tables supports a set of features from both of these applications, including visualizing, querying, and linking data tables. It excludes certain features that are highly specialized, error prone, or too cumbersome for use on a mobile device. Data tables can be viewed using a set of built-in views as well as a number of customizable views, including List, Detail, Map, and Graph views. These views are implemented in HTML/JavaScript and do not require recompilation to accommodate changes in configuration.

ACM DEV-4, Dec 06—7 2013, Cape Town, South Africa
ACM 978-1-4503-2558-5/13/12.
http://dx.doi.org/10.1145/2537052.2537077

For example, an application may consist of a table of vaccine storage facilities. Another table stores a list of refrigerators, each associated with a facility by means of the facility key as in a relational database. In ODK Tables, this data can be viewed with a built-in spreadsheet view, as if viewed in Excel. Additional custom views can also be included, so that a List view written in HTML displays an element for each row. Clicking on an element opens a Detail view, where all the information in the row is presented in a customized fashion. For instance, the types of power supply at this facility can be indicated with checkboxes, and a link can be presented in the view that opens another List view displaying all the refrigerators at the facility.

Over the past year, ODK Tables has been used to build several real-world applications, including an actual deployment. The first was a country-level vaccine cold-chain database. This includes all the vaccine storage facilities in the country, as well as all the refrigerators in the cold chain. It applies many features originally presented as part of ODK Tables, including Map views and conditional color rules. With these features an administrator is able to look at a map and easily visualize the state of the cold-chain within a country.

The second application involved setting up Tables to serve as the main data collection interface for a longitudinal HIV study. Subjects are tracked over a period of two years, with regular follow-up visits. Each follow-up corresponds to a distinct time point in the study and requires a specific form. Further, males and females require different forms, as do subjects in different age ranges.

In the course of preparing these applications it became apparent that several elements of the original Tables design were preventing the application from reaching its goal of being highly useful to a range of applications. Specifically:

1. **Disconnected configurability.** A complete ODK Tables application, including data and custom views, can be downloaded from a server. However, in many cases configuration of a device is expected to occur in an offline setting with a minimal amount of effort, often by non-technical users.
2. **Users require app-level customization.** While table-level customization in the form of a variety of views was useful, users also desired a customizable way to navigate through the application.
3. **Scalability.** Many applications involve thousands of rows and dozens of columns. To avoid memory constraints, we were forced to alter the way the data objects are passed to Android WebViews and the way complex objects are serialized to the database.

We present an overview of how we addressed the above issues, and provide examples of how the use cases motivated their resolution.

2. DISCONNECTED CONFIGURATION

Configuration of an application must be possible in the absence of a reliable data connection. Further, it must be simple and straightforward so that it can be accomplished by non-technical users. This allows the addition of devices to a deployment without requiring the presence of a study technician. This was of particular importance to the HIV study, which is taking place over a period of two years at a number of sites. Disconnected configuration allows the addition of a device to the study to be a simple process and decouples the administration of the study from the specific devices that are deployed.

Addressing this problem consisted first of allowing a complete export of data, including all metadata in a data table. An exported table will thus include all the ODK Tables-specific metadata relevant for the synchronization of each row, as well as the user-defined data. This ensures that a complete set of information, including the state of the data within the application, can be moved between phones.

Second, the application must be able to reference and import this data with minimal user involvement. This was accomplished by supporting a configuration file. At startup, it scans the base directory for a file named "config.properties", which is formatted as a conventional Java properties file. This includes information about the files to import. Thus to configure the application, a user must copy a directory containing a configuration file onto the device. When the application is opened, if the properties file has been modified, configuration occurs and the application is initialized without requiring any additional user interaction.

3. APPLICATION CUSTOMIZATION

The burden on the interviewer to select the right form can be minimized if the study designer is able to customize the entire experience with the application. While navigating a data table via a series of customized HTML skins makes the data more manageable and adds valuable functionality for our users, it is does not go far enough in providing an overall customized experience at the application level. Originally, when the ODK Tables application was opened, the user was first presented with a list of all the data tables in the database. The customized interaction with the data began after a table was selected from this list. However, some users also require context to help them choose the correct table by which to enter into the data set. This reduces the amount of training required and increases the likelihood of strict adherence to the study protocol.

The HIV study follows this model, and selection of the correct form is facilitated by application-level customization. ODK Tables now opens with a screen defined in HTML that presents options to follow up with an existing patient or enroll a new patient. At that point it begins to restrict the number of forms presented as options to the interviewer based on the study protocol, which has been encoded into the HTML skins by the study designer. Rather than expecting the interviewer to select the correct form from the entire list, they are shown only the applicable options. It is then possible to query the local database to see which visits have already been entered and thus present only the applicable forms, rather than the entire list of forms available to the application.

ODK Tables can still operate in the conventional mode described in [3], where a customized navigational experience occurs only when viewing a data table. However, at startup it now first checks to see if a custom home screen file, written in HTML/JavaScript, has been specified. If it has been, the application presents this customized page rather than the default list of data tables. As with the other custom views in ODK Tables, this home screen can take complete advantage of all the tools available in HTML and JavaScript.

4. SCALABILITY

Both the cold-chain and HIV study applications described above included larger datasets than were used in the original studies. These include tables with thousands of rows and with hundreds of columns. This produced two issues with scalability we were forced to address.

The first was the way that Java objects were passed to the Android WebView that is responsible for rendering the customized HTML/JavaScript files. An object representing a data table is given to the WebView as a JavaScript interface with the handle "data". An object is added each time a table is viewed, so that 10 data objects are added to the WebView over the course of viewing 10 tables. It was discovered that the despite the old objects ostensibly being unreachable, as "data" in the JavaScript refers only to the latest table, the objects were not being garbage collected. If a number of large tables were viewed, this resulted in an eventual crash as the JVM memory limit was exceeded. This was of particular concern in the cold-chain application, where a user might browse a list of all the refrigerators in a country inventory numerous times in a single ODK Tables session. We now instead pass a wrapper containing a WeakReference to the object and keep the strong reference in the Android code, thus allowing appropriate garbage collection.

Second, we persist many of the more complex objects in our data model to the database as JSON serializations. Dealing with a number of tables, many with hundreds of columns, resulted in a decrease in performance as the objects were de- and re- serialized. This was completely mitigated by implementing a more aggressive caching system for our own abstraction objects.

5. CONCLUSION

These applications demonstrate the capabilities of ODK Tables as a data browser and manager. They have also highlighted the importance of functioning in disconnected environments as well as providing a means to completely customize interactions with data. Real-world evaluations of the tool have motivated the evolution of Tables away from being simply a data manager. With the advent of complete application-level customization, it has become a framework for designing mobile applications with web-based tools that do not require recompilation. A single installation of ODK Tables can thus be used with a variety of HTML skins in order to produce a number of distinct data-driven applications for disconnected environments.

6. REFERENCES

[1] C. Hartung, Y. Anokwa, W. Brunette, A. Lerer, C. Tseng, G. Borriello. "Open Data Kit: Building Information Services for Developing Regions." In Proc. of 4th IEEE/ACM Information & Communication Technologies for Development (ICTD 10), Dec 2010.

[2] Open Data Kit. http://opendatakit.org/

[3] Waylon Brunette, Samuel Sudar, Nicholas Worden, Dylan Price, Richard Anderson, and Gaetano Borriello. 2013. "ODK tables: building easily customizable information applications on Android devices." In Proc. of the 3rd ACM Symposium on Computing for Development (ACM DEV 13), Jan 2013.