Sterling Survey System

Designing the Server, Database, and Administrative Tools in Support of a Multi-Mode, Multi-Platform Survey Authoring Framework

Chris Schlapper
University of Wisconsin Survey Center
University of Wisconsin-Madison

2017 International Field Directors and Technologies Conference
May 23, 2017
Sterling Survey System

- Survey Authoring Framework
  - All survey modes
  - Multi-Mode
  - Multiple Platforms and devices
Server-Side Implementation

• Server configuration
  • Server Framework
  • Database
  • Survey administrative tools
Server-Side Implementation

- Server configuration
- Server Framework
- Database
- Survey administrative tools
Command Server
Server Framework Technology – Node.js

- Open-Source JavaScript runtime environment
  - Execute JavaScript code outside of a browser
- Open-Source libraries
  - Extend capabilities of standard JavaScript
- Examples
  - https
  - Jquery
  - Bluebird
  - JsonToCsv
Why Node.js?

- JavaScript, same software language used for instrument authoring
- Large open-source libraries
- Ability to build a targeted web communication protocol
  - Full web-server functionality not desired
  - Only small set of valid commands needed
Command Server Functions

- Programmed using Node.js
- Validates client-issued commands
  - JSON data structure containing parameters
  - Invalid commands or parameters are not processed
- Executes allowed commands
- Communicates with database
  - User account with limited access
  - Updates
  - Queries data to return to client device
Command Server – Three Commands

• Save cases to database
  • Collected by the instrument
  • All data not previously saved

• Synchronize
  • Adds or removes assigned cases on client device
  • Updates case ownership in database

• Save LocalStorage
  • Copies client device’s LocalStorage to database
  • Backup and debugging resource
Server-Side Implementation

- Server configuration
- Server Framework
- Database
- Survey administrative tools
Server Database Technology – NoSQL Database

• MongoDB
  • Stores documents, not records
  • Associates data in complex ways
  • JSON data structure
    • Key/Value pairs
    • Nested sub-documents
      • As many nested sublevels as you need
      • Rosters
  • Flexibility in database design
  • Data mining
Why Not SQL Database?

- Relational model
  - MySQL
    - Familiarity
    - Wide-adoption
  - Strict schema definitions
    - Changes during field period
    - Schema changes over time
      - Querying across multiple projects may fail
Why Not SQL Database?

- **Entity-Attribute-Value (EAV) model**

<table>
<thead>
<tr>
<th>csid</th>
<th>q1</th>
<th>q2</th>
<th>q3</th>
<th>q4</th>
<th>q5</th>
<th>q6</th>
</tr>
</thead>
<tbody>
<tr>
<td>113489</td>
<td>Y</td>
<td>15</td>
<td>10/31/2016</td>
<td>N</td>
<td>DK</td>
<td>14</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>entity</th>
<th>attribute</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>113489</td>
<td>q1</td>
<td>Y</td>
</tr>
<tr>
<td>113489</td>
<td>q2</td>
<td>15</td>
</tr>
<tr>
<td>113489</td>
<td>q3</td>
<td>10/31/2016</td>
</tr>
<tr>
<td>113489</td>
<td>q4</td>
<td>N</td>
</tr>
<tr>
<td>113489</td>
<td>q5</td>
<td>DK</td>
</tr>
<tr>
<td>113489</td>
<td>q6</td>
<td>14</td>
</tr>
</tbody>
</table>

- Provides flexibility
- Writing queries more challenging
  - Search the attributes to understand the data structure
- Slow query response time
  - Pivot tables, GROUP_CONCAT
Why MongoDB?

• Non-Relational database
  • No mandatory schema definitions
    • Allow for changes during field period
  • Implement standard variables
    • Allow for querying across multiple projects without breaking for older projects
• Nested sub-documents
Why MongoDB?

- Big Data
  - Indices
  - Fast query-response time
  - Scale horizontally
    - Distribute data across multiple machines
- JSON data structure
  - Native to JavaScript/node.js
  - Client-side and server-side
Node.js Implementation Notes

- Node Package Manager (npm)
  - Install libraries
    - Some libraries depend on other libraries
    - Global or local installation
- Asynchronous code execution
  - Ideal for client-side website
  - Challenges on server-side
  - Promises
    - Bluebird module
MongoDB Notes

• No security “out-of-the-box”
• NoSQL query language
  • Different from SQL
  • Retrieving data from sub-documents
• RoboMongo
  • Administrative console program
MongoDB Notes – Database Design

• Optimize for NoSQL
• No table joins
  • Run multiple queries and programmatically join the data
  • Data-duplication recommended
    • Store “join” data as sub-document
• JSON structure
  • Associative arrays
  • Avoid using a value for the key
    • Optimized for client-side
    • Difficult to query on database
      • Key analogous to a database column name
Server-Side Implementation

- Server configuration
- Framework
- Database
- Survey administrative tools
Administrative Requirements

- Used by Project Directors, Programmers, and Field Supervisors
- Save Question data
  - During survey authoring process
- Instrument debugging
- Load sample
- Manage project during field period
  - Data dashboard and reports
  - Case assignment
  - Roll-back case to a prior version
- Data delivery
Admin Server Functions

• Programmed using Node.js
• Separate path from client-issued commands
  • Behind firewall
  • More functionality
• JSON data structure containing parameters
• Invalid commands or parameters are not processed
• Requires greater security permissions
• Executes allowed commands
• Communicates with database
  • User account with elevated access
    • Create new schema
    • Access to administrative data collections
Admin Server – Commands

• Save Question data during authoring process

• Load sample data

• Create data delivery files

• Fulfill queries for Project Management software
Monocle 3.0
### Summary

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sample</td>
<td>844</td>
</tr>
<tr>
<td>Worked</td>
<td>844</td>
</tr>
<tr>
<td>Complete</td>
<td>843</td>
</tr>
<tr>
<td>Non-Complete</td>
<td>7</td>
</tr>
<tr>
<td>Ineligible</td>
<td>83</td>
</tr>
<tr>
<td>Partial</td>
<td>0</td>
</tr>
<tr>
<td>Response Rate</td>
<td>0</td>
</tr>
</tbody>
</table>

### Completes by Gender

<table>
<thead>
<tr>
<th>Gender</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>420</td>
</tr>
<tr>
<td>M</td>
<td>423</td>
</tr>
<tr>
<td>Unknown</td>
<td>0</td>
</tr>
</tbody>
</table>

### Q3 Summary

<table>
<thead>
<tr>
<th>Group</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC</td>
<td>753</td>
</tr>
<tr>
<td>I1</td>
<td>17</td>
</tr>
<tr>
<td>I2</td>
<td>45</td>
</tr>
<tr>
<td>I3</td>
<td>12</td>
</tr>
<tr>
<td>I4</td>
<td>1</td>
</tr>
<tr>
<td>I5</td>
<td>6</td>
</tr>
<tr>
<td>I7</td>
<td>2</td>
</tr>
<tr>
<td>N1</td>
<td>2</td>
</tr>
<tr>
<td>N2</td>
<td>3</td>
</tr>
<tr>
<td>N4</td>
<td>2</td>
</tr>
</tbody>
</table>

### Q8 Summary

<table>
<thead>
<tr>
<th>Group</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>54</td>
</tr>
<tr>
<td>N</td>
<td>699</td>
</tr>
<tr>
<td>I1</td>
<td>17</td>
</tr>
<tr>
<td>I2</td>
<td>45</td>
</tr>
<tr>
<td>I3</td>
<td>12</td>
</tr>
<tr>
<td>I4</td>
<td>1</td>
</tr>
<tr>
<td>I5</td>
<td>6</td>
</tr>
<tr>
<td>I7</td>
<td>2</td>
</tr>
<tr>
<td>N1</td>
<td>2</td>
</tr>
<tr>
<td>N2</td>
<td>3</td>
</tr>
<tr>
<td>N4</td>
<td>2</td>
</tr>
</tbody>
</table>
### Assign Selected Cases to:
- Schlapper, Chris (2679)

**Show 10 entries**

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Last ID</th>
<th>Accessed</th>
<th>Version</th>
<th>Code</th>
<th>Attempt Success</th>
<th>Type</th>
<th>Stratum</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1002011</td>
<td>16</td>
<td>5</td>
<td>1</td>
<td>1100</td>
<td>12</td>
<td>Cig</td>
<td>3</td>
<td>2016-07-03T15:13:03.068Z</td>
</tr>
<tr>
<td>1002016</td>
<td>16</td>
<td>6</td>
<td>1</td>
<td>1100</td>
<td>1</td>
<td>Cig</td>
<td>3</td>
<td>2016-06-23T16:02:00.794Z</td>
</tr>
<tr>
<td>1002027</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1100</td>
<td>1</td>
<td>E-Cig</td>
<td>3</td>
<td>2016-06-23T17:24:44.412Z</td>
</tr>
<tr>
<td>1002049</td>
<td>16</td>
<td>1</td>
<td>1</td>
<td>1100</td>
<td>1</td>
<td>Cig</td>
<td>3</td>
<td>2016-06-23T18:04:21.248Z</td>
</tr>
<tr>
<td>1002063</td>
<td>16</td>
<td>2</td>
<td>1</td>
<td>1100</td>
<td>1</td>
<td>Cig</td>
<td>3</td>
<td>2016-06-23T16:22:26.594Z</td>
</tr>
</tbody>
</table>
1. Create a sample.csv file for your project.
2. Copy that file into the c:\Robot\sample\ directory
3. Press the button.

Begin importing sample for p1156
• Current implementation produces
  – CSV file
  – SPSS syntax file
Monocle 3.0 - Future Features

• Login authentication
  • Menu of authorized projects
• Setup screens
  • Interviewers
  • Summary screen
  • Caselist
• Expand data delivery features
  • User-select subset of records and variables
  • Partial data delivery
• Implement current reports
Server-Side Implementation

• Server configuration
  • Data Entry
  • CATI
  • Web
• Framework
• Database
• Survey administrative tools
Admin Tools
CATI and Data Entry
Web Server
Admin Server
Command Server
MongoDB Server

CAPI
Private
Public
Admin Tools

CATI and DE

Web Server

Admin Server

Command Server

MongoDB Server

CAPI

Web Server

Web
Server-Side - Future Features

• Existing processes
  • Integration
    • Rapid implementation
  • Redesign
    • Optimize use of NoSQL
      • Data structure
      • Shard data on multiple machines
  • Examine and implement new efficiencies
Acknowledgements

- Steve Bochte
- Brendan Day
- Jaime Faus
- Kate Krueger
- Marie Nitschke
- Augie Salick
- Eric White
Tech Showcase – Session 10F

Join us for a hands-on demonstration!

Tech Showcase
3:15 pm – 4:30 pm
Thank You!

For copies of this presentation or more information, contact:

Chris Schlapper
cschlapper@ssc.wisc.edu

Please visit us at:
www.uwsc.wisc.edu