Document owner: Angela Doxsey
Last Modified: 8/23/2013

Managing Interoperability between the OpenHDS Server and Android Tablet Application

I. Introduction

a. The Incentive

The inspiration behind this Master's project was to develop a proficient understanding of the communication between a web server and an Android tablet application. The approach I chose was to investigate the OpenHDS application (https://code.google.com/p/openhds/), which maintains a consistent record of significant demographic events that occur to a population in a fixed geographic region. The OpenHDS application includes an Android tablet device application that facilitates the collection of demographic and health data in developing countries. It aims to collect longitudinal demographic data to identify population dynamics such as mortality trends and migration rates. The data is eventually used by researchers to identify the most common causes of fertility, morbidity, and mortality within a given population.

b. The Result

The culmination of this project is an easily managed interface in the OpenHDS tablet application that provides implementers of OpenHDS with the ability to customize their location hierarchies. The incentive for this functionality is that location hierarchies are defined and interpreted differently across regions of the world; for example, the U.S. may pinpoint a location based on five hierarchy levels: Country/Region/State/County/Town. In Ethiopia, there may be four hierarchy levels: Region, Sub-region, Village, and Sub-village With the new functionality that was developed in this project, an implementer can anywhere between one to nine location hierarchies. The naming of these hierarchies requires no computer programming experience. It is established via an accessible user interface on the tablet application.

c. The Path

The first step in achieving the knowledge required to develop this new functionality for the OpenHDS application was to study the Android operating system, particular in the context of tablet development. Next, the OpenHDS server application was inspected. This included establishing communication with a MySQL database. It also included developing an understanding of the inner workings of the OpenHDS server's use of Hibernate (http://www.hibernate.org/) to facilitate the storage and retrieval of Java domain objects via Object/Relational Mapping. Finally, the current functionality the OpenHDS tablet application was explored. This led to the development of the code that today provides the customizable location hierarchies discussed above.

II. Getting to Know Android

a. Why Android?

Throughout my graduate degree, I was fascinated by mobile application development and came to believe that it was the way of the future. I knew that in order to feel like my computer science degree was complete, I needed to know something about mobile development. In my eyes, this meant either Apple or Android. I didn't want Apple for personal reasons. Android is a platform that enticed me with its Google, Java, and Linux backbone. It provides the core building blocks for an efficient architecture, manageable user interfaces, and complex communication with both internal and external components.

b. The Process

While I have chosen a career that will require that I use a computer all day, I am a book reader. I like to turn pages, not scroll screens. My first step in getting to know Android was to read - essentially cover-to-cover - the text Android in Practice by Charlie Collins, Michael Galpin, and Matthias Kappler. This text included many Android code examples that introduced me to the key mechanisms I needed to know: how to build user interfaces, share data between applications, communicate via HTTP & web services, store and retrieve data, and more. This text proved an excellent resource for the core functionality of Android. This being said, the most current Android documentation is available online. I supplemented reading the textbook with example online tutorials, Stack Overflow searches, and peer inquiries.

c. The Challenge

The toughest part about Android was not the new structure of coding with aspects like Activities and Intents, but rather the build and debug environment. I knew that with OpenHDS in the near future, I'd need to learn how to build with Maven. This required making Eclipse/Java Development Tools (JDT), the Android Toolkit (ADT), and the maven-eclipse-plugin work in harmony with each other. Eclipse allowed for core Java tooling and compiling, the ADT for packaging APKs (Android application package files), and Maven to allow for dependency resolution. Integrating these components took patience and an abundance of the Stack Overflow searches mentioned above. The debug environment was also new. I had never needed to work with multiple hardware devices (i.e. tablet and computer) in debugging code. Getting the setup just right for this, with the correct versions of the Android SDK and the Android Debug Bridge, was a challenge.

III. Getting to Know OpenHDS

a. The Challenge (The Process)

The approach I took with OpenHDS allowed me to articulate what I believe to be my greatest weakness as a software developer. In the second month of the project, I plunged into my first code development task in the OpenHDS tablet application. I wrote a mouthful of Android code that I would eventually chew and spit out when I got my head back on straight. Afterward, I took a step back and decided that Android needed to be put on the shelf. I became familiar with the OpenHDS server application itself. I had been trying to build an application to communicate with a server when I didn't know what the server was doing. I learned that I need to lay out a plan of attack for myself. This lesson will drive my future as a developer.

After this hiccup, I took a bird's eye view of the OpenHDS server. I set it up with a MySQL database and got it working properly with Hibernate. I kept digging into the server code until I was eventually able to understand it and operate it properly, which meant to first sync its data onto an existing version of the tablet application. This required gaining knowledge about some of the quirks surrounding tablet/server communication, including configuring static IPs. I then worked with the existing version of the tablet application until it would upload forms via ODK Collect (http://opendatakit.org/use/collect/) and communicate effectively with the existing version of the OpenHDS server.

<u>IV. Writing the Android Code</u>

a. The Reward (The Process)

Once I became proficient in how the OpenHDS server and tablet were coexisting, adding new functionality on the tablet side was relatively straightforward. The sample Android applications that I had developed in isolation of a server acted as a foundation of the new code I needed to develop. Because I had already dug deep into the way that the OpenHDS server communicated with its current mobile counterpart, many of the obstructions I experienced in developing new code looked familiar: I had already solved them in trying to make the pre-existent code functional. The code I developed took on a rhythm that acted as my reward of drowning line-by-line in the Eclipse debugger the month prior. I finally was able to swiftly develop the Android code and produce the functionality that I had originally desired.

<u>V. Conclusion</u>

In the very beginning stages of this project, I thought I wanted to develop a flashy Android application that I could show off in the Android market. Instead, I entered a detailed study of a server, surveyed the way it used ODK Collect to provide form upload, provided persistence to it via a MySQL database, and contributed new functionality for its two-way communication with a tablet. Somewhere in this process, perhaps when I was finally allowed to come up for air from underneath the blinding rays of the Eclipse debugger, I recognized something about software development: interoperability makes it interesting. I knew from the get-go that I loved software development because it was challenging and drove me a little crazy. This Master's project has helped me to recognize why exactly I know this feeling will never change. As software developers, all we need to do to make our lives more exciting is to get more devices to talk to one another. The possibilities for working with interoperable components are endless. We allowed remotes to talk to TVs, bikes with odometers, sunglasses with cameras, cell phones with databases of music...software that talks to other software is interesting software.

Appendix A. OpenHDS Custom Location Hierarchy Documentation

a. Introduction

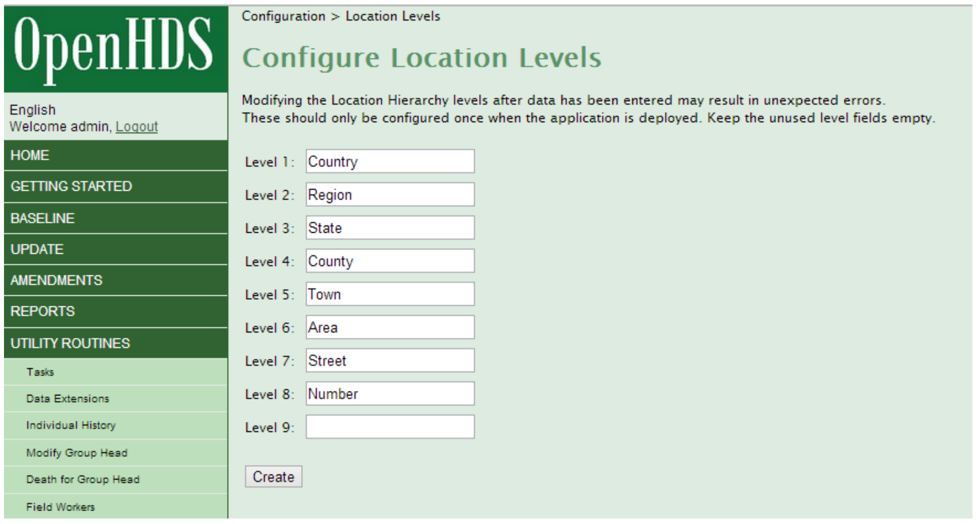
Appendix A provides an explanation to configure your OpenHDS Tablet application to adhere to a custom Location Hierarchy. This document assumes that you are familiar with the OpenHDS web application and its associated Wiki, available from https://code.google.com/p/openhds /wiki/WelcomePage?tm=6.

The OpenHDS Tablet application now provides an option to accommodate anywhere from one to nine location hierarchy levels. This complements the capability already provided in the OpenHDS web application. In order to implement this new functionality, you must apply your configuration to both the server and the tablet.

b. Configuring the OpenHDS Server Location Hierarchies

As is explained in the OpenHDS Wiki, the server provides configuration for Location Levels via the below interface. As an example, I have created a Location Hierarchy with 8 levels. These Location Hierarchy will act as an effective example for the remainder of this guide.

http://your-ip:8080/openhds/location-level



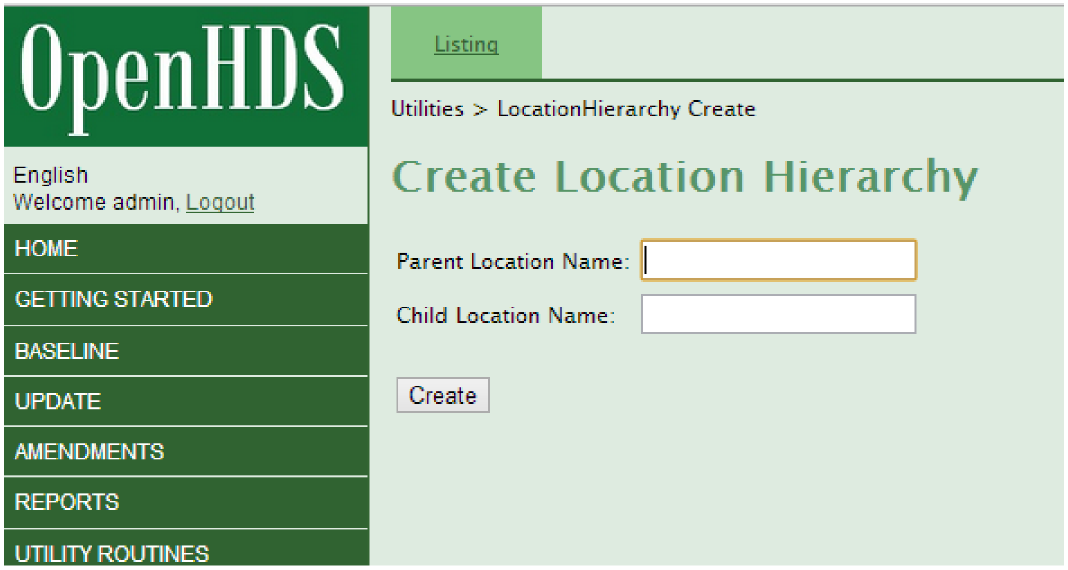
Configure Location Levels

Modifying the Location Hierarchy levels after data has been entered may result in unexpected errors.
These should only be configured once when the application is deployed. Keep the unused level fields empty.

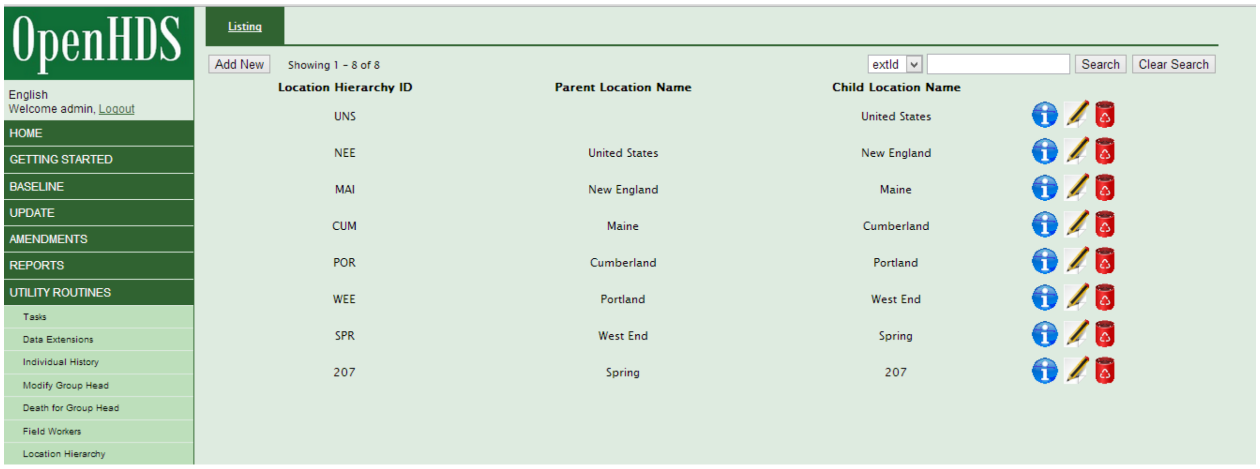
Level 1:	Country
Level 2:	Region
Level 3:	State
Level 4:	County
Level 5:	Town
Level 6:	Area
Level 7:	Street
Level 8:	Number
Level 9:	

As is instructed in the OpenHDS 'Getting Started' guide (http://your-DNS:8080/openhds/getting-started/index.faces), once the location levels have been created, you must create your own Location Hierarchies via the following URL:

http://your-ip:8080/openhds/locationhierarchy/create.faces



Under "Listing":



Location Hierarchy ID	Parent Location Name	Child Location Name
UNS		United States
NEE	United States	New England
MAI	New England	Maine
CUM	Maine	Cumberland
POR	Cumberland	Portland
WEE	Portland	West End
SPR	West End	Spring
207	Spring	207

Note that the above configuration follows the following Location Level/Hierarchy pattern:

Level	Level Identifier	Location Hierarchy	Hierarchy Identifier
Level 1	Country	United States	UNS
Level 2	Region	New England	NEE
Level 3	State	Maine	MAI
Level 4	County	Cumberland	CUM
Level 5	Town	Portland	POR
Level 6	Area	West End	WEE
Level 7	Street	Spring	SPR
Level 8	Number	207	207

Once you have configured the OpenHDS web application, you can now configure your tablet to accommodate the Location Hierarchy.

Configuring Custom Tablet Location Hierarchies

I. Complete the initial tablet setup.

The first steps are outlined in the OpenHDS Tablet wiki (https://code.google.com/p/openhds/wiki/TabletSetup). These steps are to:

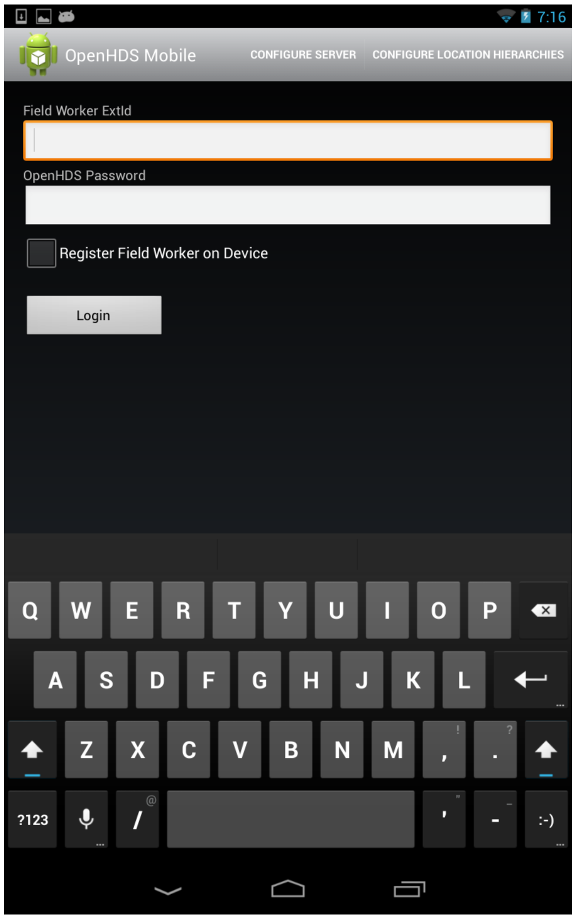
- Select **Log in as Field Worker**
- Select **Configure Server** in the action bar, and set the **OpenHDS Web Service** URL to point at your OpenHDS installation. Also, provide a username and password to be used for downloading data.
- Enter in a **Field Worker ExtId** that is registered in the OpenHDS. On first login, enter a password to use for the field worker. Select **Register Field Worker on Device**, and hit

Register.

- If successful, uncheck **Register Field Worker on Device** and select **Login**.

II. Configure your custom Location Hierarchies.

- Navigate to the Login screen, and select **Configure Location Hierarchies** from the action bar.

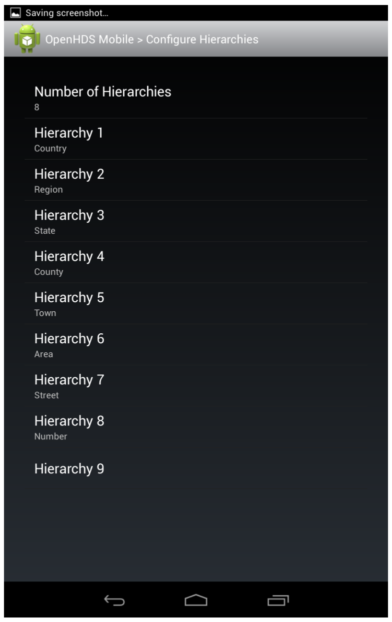


- You will see a list of options. Select the first to indicate the number of Location Levels that you configured prior in the OpenHDS web application.

- Once you have selected the number of Location Levels, select each subsequent level individually to change its Level Identifier on the tablet. These must reflect the Level Identifiers that you configured prior in the OpenHDS web application.

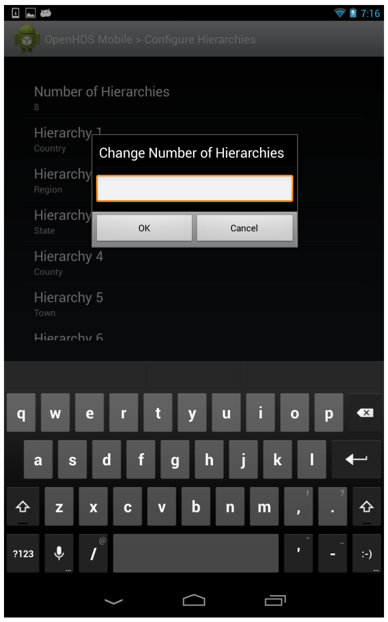
- **DO NOT FRET** if you do not see your changes immediately reflected on this interface! The application is aware that you have changed it, and the changes will be reflected on the interface when the page is refreshed.

  There is no refresh button provided because there is currently no clean way to refresh preferences in Android. If you find that it isn't clear what you have changed on this interface, hit the **Back arrow and then click **Configure Location Hierarchies** again to see your changes. I recognize that this is a roundabout way of completing this step, but we can hope that there will be a clean way to refresh preferences in newer versions of Android.



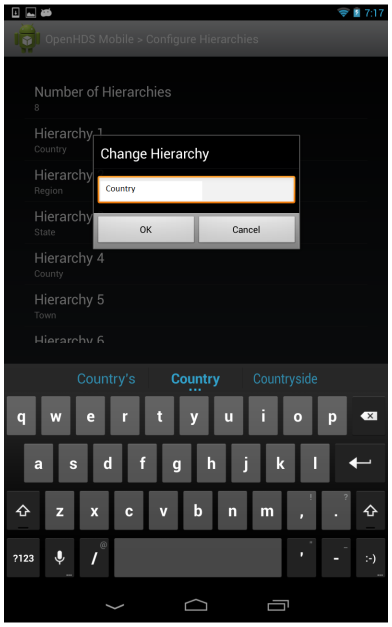
Saving screenshot...
OpenHDS Mobile > Configure Hierarchies

Number of Hierarchies
8
Hierarchy 1
Country
Hierarchy 2
Region
Hierarchy 3
State
Hierarchy 4
County
Hierarchy 5
Town
Hierarchy 6
Area
Hierarchy 7
Street
Hierarchy 8
Number
Hierarchy 9



OpenHDS Mobile > Configure Hierarchies

Change Number of Hierarchies

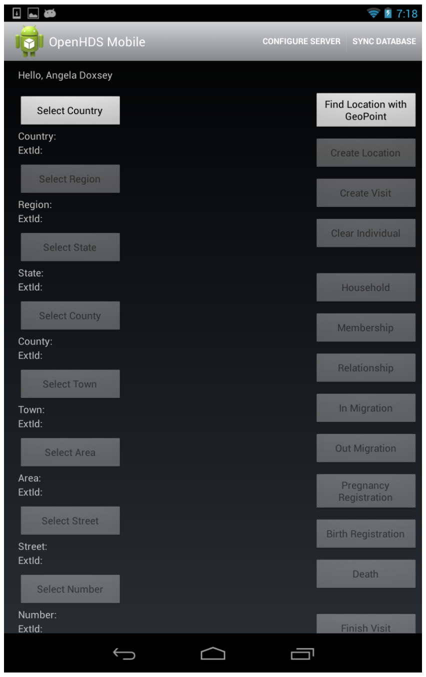


OpenHDS Mobile > Configure Hierarchies

Change Hierarchy
Country

III. Start using the tablet with your custom Location Hierarchies.

- Login as a field worker with your username and password. After logging in, select **Sync Database** from the action bar. Proceed to sync database.
- Now, when you login, you will be brought to an interface that reflects the changes you made. Navigate through this interface, selecting the Location Hierarchies that were downloaded to the tablet when you synced the database.



Hello, Angela Doxsey

Country:
ExtId:

Region:
ExtId:

State:
ExtId:

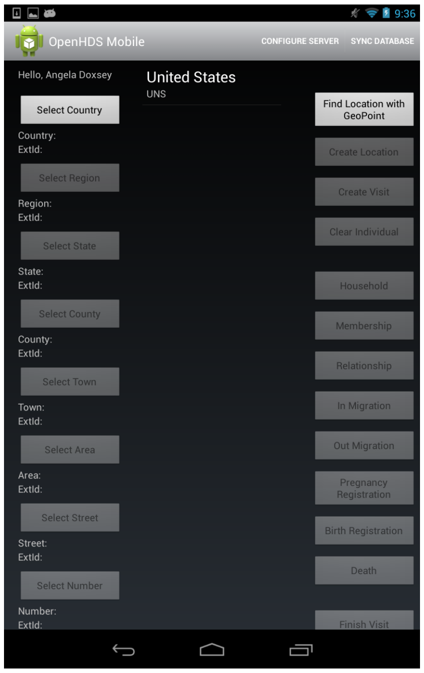
County:
ExtId:

Town:
ExtId:

Area:
ExtId:

Street:
ExtId:

Number:
ExtId:



Hello, Angela Doxsey

United States
UNS

Country:
ExtId:

Region:
ExtId:

State:
ExtId:

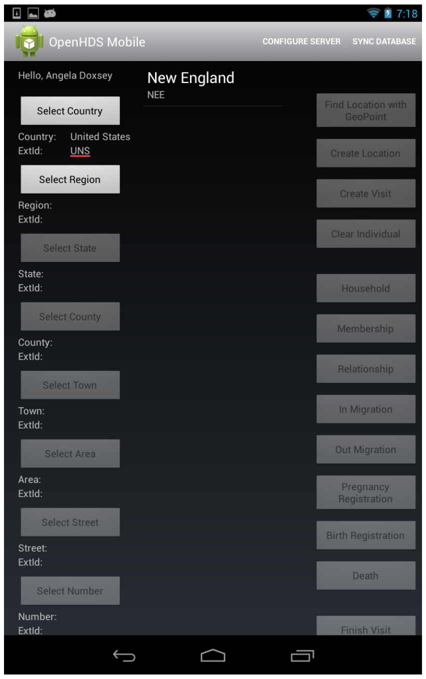
County:
ExtId:

Town:
ExtId:

Area:
ExtId:

Street:
ExtId:

Number:
ExtId:



Hello, Angela Doxsey

New England
NEE

Country: United States
ExtId: UNS

Region:
ExtId:

State:
ExtId:

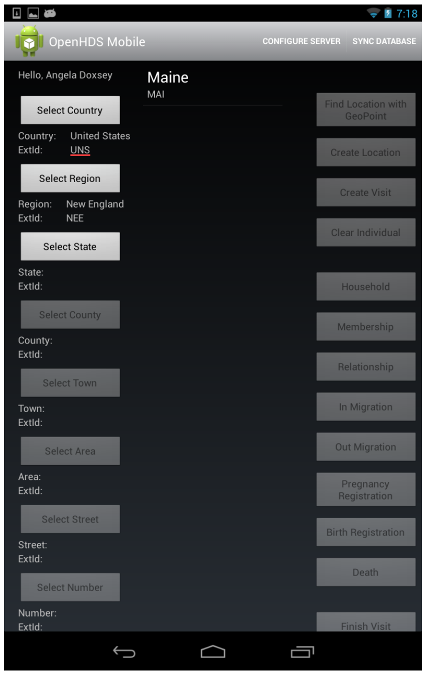
County:
ExtId:

Town:
ExtId:

Area:
ExtId:

Street:
ExtId:

Number:
ExtId:



Hello, Angela Doxsey

Maine
MAI

Country: United States
ExtId: UNS

Region: New England
ExtId: NEE

State:
ExtId:

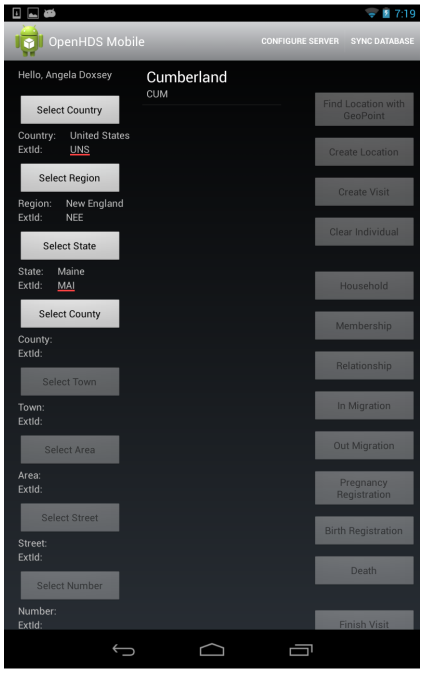
County:
ExtId:

Town:
ExtId:

Area:
ExtId:

Street:
ExtId:

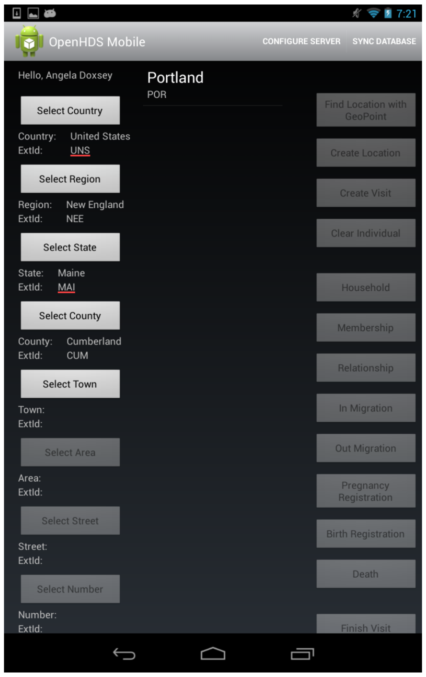
Number:
ExtId:



Hello, Angela Doxsey

Cumberland
CUM

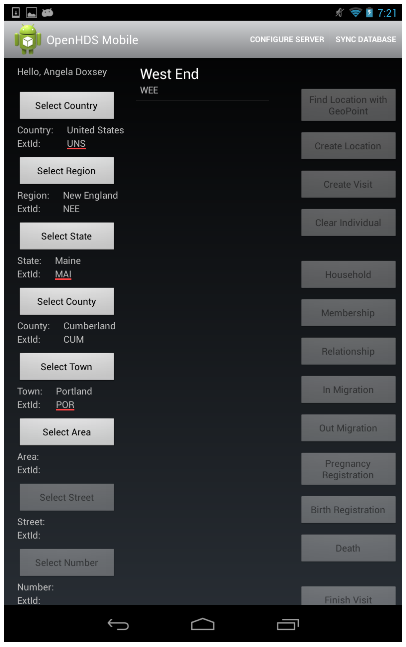
Country: United States
ExtId: UNS
Region: New England
ExtId: NEE
State: Maine
ExtId: MAI
County:
ExtId:
Town:
ExtId:
Area:
ExtId:
Street:
ExtId:
Number:
ExtId:



Hello, Angela Doxsey

Portland
POR

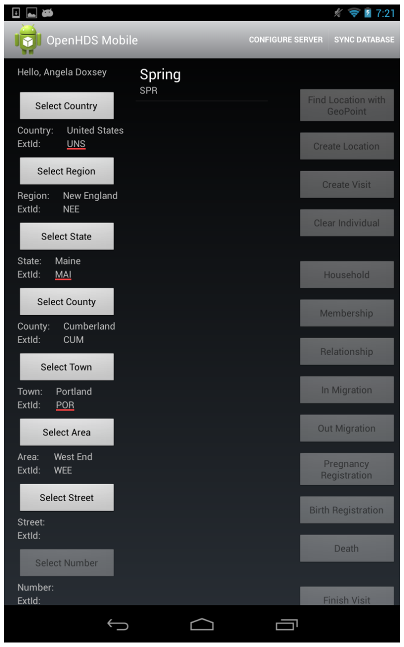
Country: United States
ExtId: UNS
Region: New England
ExtId: NEE
State: Maine
ExtId: MAI
County: Cumberland
ExtId: CUM
Town:
ExtId:
Area:
ExtId:
Street:
ExtId:
Number:
ExtId:



Hello, Angela Doxsey

West End
WEE

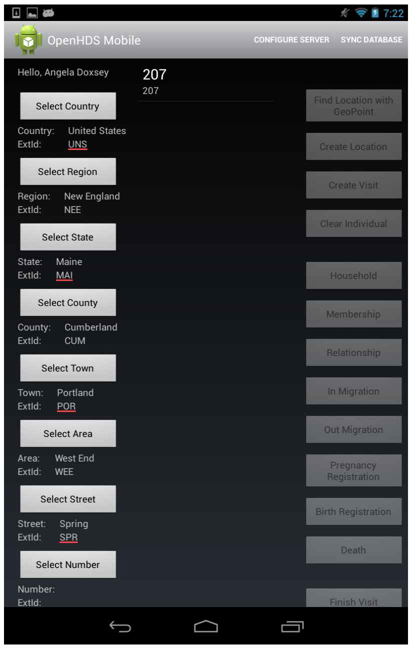
Country: United States
ExtId: UNS
Region: New England
ExtId: NEE
State: Maine
ExtId: MAI
County: Cumberland
ExtId: CUM
Town: Portland
ExtId: POR
Area:
ExtId:
Street:
ExtId:
Number:
ExtId:



Hello, Angela Doxsey

Spring
SPR

Country: United States
ExtId: UNS
Region: New England
ExtId: NEE
State: Maine
ExtId: MAI
County: Cumberland
ExtId: CUM
Town: Portland
ExtId: POR
Area: West End
ExtId: WEE
Street:
ExtId:
Number:
ExtId:



Hello, Angela Doxsey

207
207

Country: United States
ExtId: UNS
Region: New England
ExtId: NEE
State: Maine
ExtId: MAI
County: Cumberland
ExtId: CUM
Town: Portland
ExtId: POR
Area: West End
ExtId: WEE
Street: Spring
ExtId: SPR
Number:
ExtId:

Everything should now be set up for you to fill out forms for the Location Hierarchy that you originally established in the OpenHDS web application.

Appendix B.

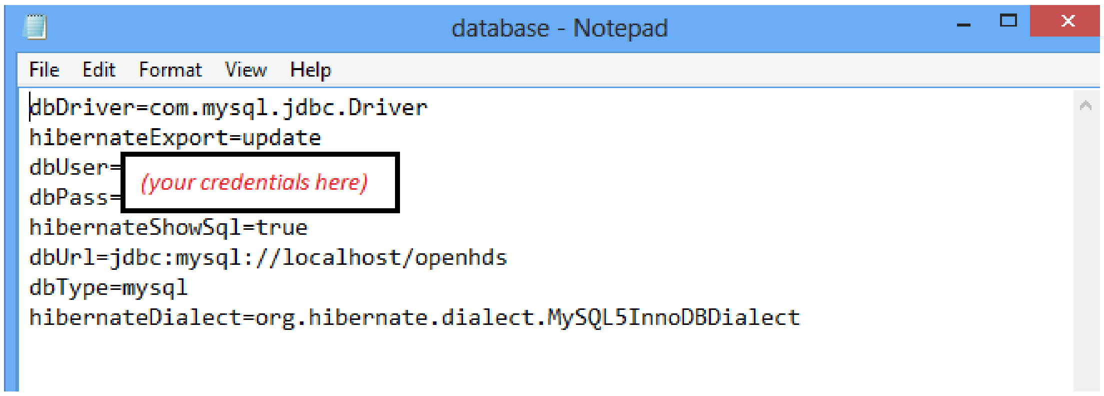
OpenHDS Database Configuration Tips

In investigating the way the OpenHDS server interacts with MySQL and Hibernate, I encountered some roadblocks that could have been easily avoided if I had some prior knowledge about Hibernate. Because other developers working with OpenHDS may come from a background with similar lacking experience, I developed this guide to supplement that already present on the OpenHDS wiki (https://code.google.com/p/openhds/wiki/DeveloperGuide#Database_Access)

The database configuration file for OpenHDS will be located in the WEB-INF\classes directory of your OpenHDS Tomcat deployment, i.e.

C:\...\Apache\...\apache-tomcat-6.0.37\webapps\openhds\WEB-INF\classes\database.properties

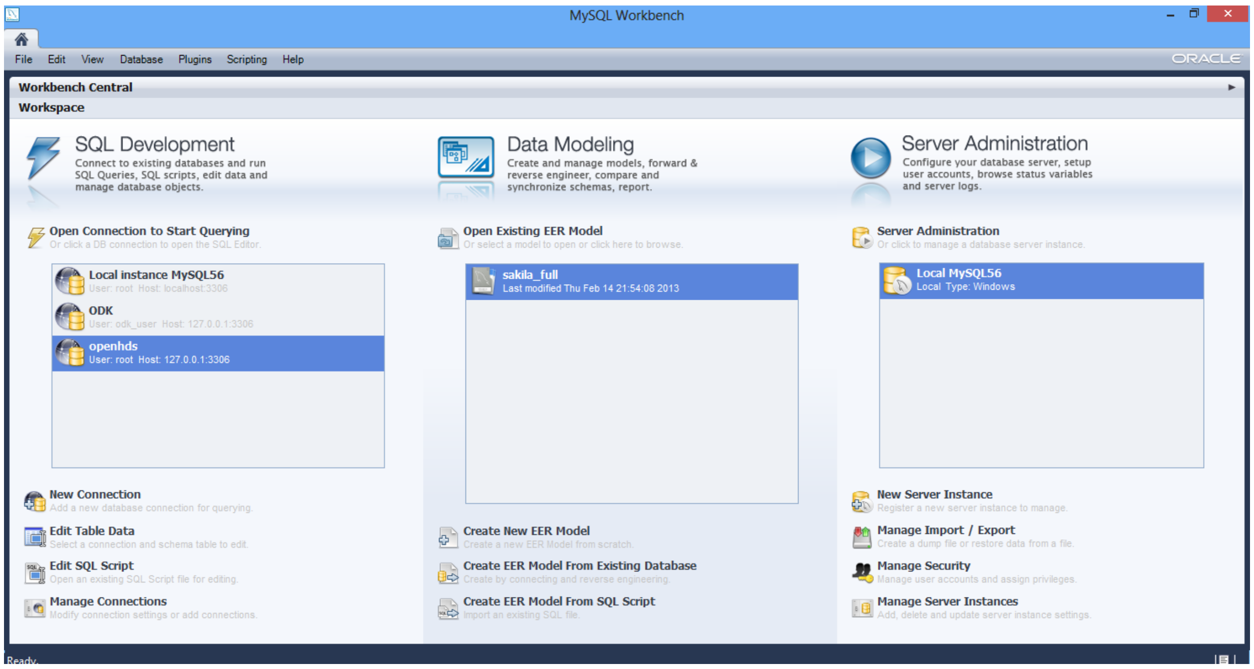
OpenHDS uses Hibernate to provide relational persistence for its data. When setting up the properties for your database, the OpenHDS wiki provides a few different options. To configure this to work most effectively with the tablet application, your database.properties file should reflect the following configuration:



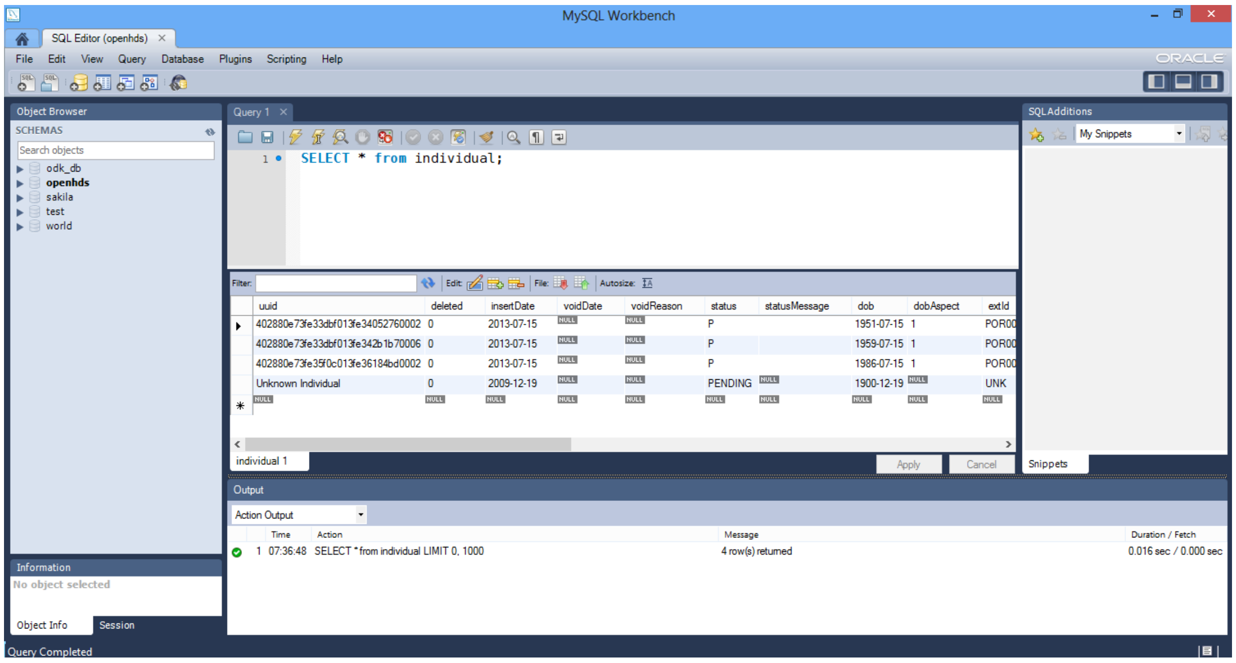
```
dbDriver=com.mysql.jdbc.Driver
hibernateExport=update
dbUser=
dbPass=
hibernateShowSql=true
dbUrl=jdbc:mysql://localhost/openhds
dbType=mysql
hibernateDialect=org.hibernate.dialect.MySQL5InnoDBDialect
```

If you adapt the above properties, you will be able to work with the OpenHDS directly. It is especially important to specify **update** as opposed to **create** for your **hibernateExport** property. If you do not do so, the data that you enter into OpenHDS will not persist beyond each new deployment. Every time you want to work with your tablet, you will need to re-enter a Baseline into OpenHDS.

I chose to interact with the OpenHDS MySQL database via MySQL Workbench:


MySQL Workbench
File Edit View Database Plugins Scripting Help
Workbench Central
Workspace
SQL Development
Connect to existing databases and run SQL Queries, SQL scripts, edit data and manage database objects.
Data Modeling
Create and manage models, forward & reverse engineer, compare and synchronize schemas, report.
Server Administration
Configure your database server, setup user accounts, browse status variables and server logs.
Open Connection to Start Querying
Or click a DB connection to open the SQL Editor.
Local instance MySQL56
User: root Host: localhost:3306
ODK
User: odk_user Host: 127.0.0.1:3306
openhds
User: root Host: 127.0.0.1:3306
Open Existing EER Model
Or select a model to open or click here to browse.
sakila_full
Last modified Thu Feb 14 21:54:08 2013
Server Administration
Or click to manage a database server instance.
Local MySQL56
Local Type: Windows
New Connection
Add a new database connection for querying.
Edit Table Data
Select a connection and schema table to edit.
Edit SQL Script
Open an existing SQL Script file for editing.
Manage Connections
Modify connection settings or add connections.
Create New EER Model
Create a new EER Model from scratch.
Create EER Model From Existing Database
Create by connecting and reverse engineering.
Create EER Model From SQL Script
Import an existing SQL file.
New Server Instance
Register a new server instance to manage.
Manage Import / Export
Create a dump file or restore data from a file.
Manage Security
Manage user accounts and assign privileges.
Manage Server Instances
Add, delete and update server instance settings.
Ready.

I was able to monitor updates that I made through the web interface to insure that the database was being updated properly. I did so by entering direct SQL queries into MySQL Workbench:



```
SELECT * from individual;
```

	uuid	deleted	insertDate	voidDate	voidReason	status	statusMessage	dob	dobAspect	extId
▶	402880e73fe33dbf013fe34052760002	0	2013-07-15	NULL	NULL	P		1951-07-15	1	POR00
	402880e73fe33dbf013fe342b1b70006	0	2013-07-15	NULL	NULL	P		1959-07-15	1	POR00
	402880e73fe35f0c013fe36184bd0002	0	2013-07-15	NULL	NULL	P		1986-07-15	1	POR00
	Unknown Individual	0	2009-12-19	NULL	NULL	PENDING	NULL	1900-12-19	NULL	UNK
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
     Time      Action                                  Message            Duration / Fetch
✓ 1  07:36:48  SELECT * from individual LIMIT 0, 1000  4 row(s) returned  0.016 sec / 0.000 sec
```

You may choose to interact in the same manner or via the MySQL commandline. Documentation for both of these options is available from http://www.mysql.com/.