**Freeminders Parse Development Guide**

A new application in parse, “freeminders-rlh” was created that heavily implements Parse relationships in an effort to implement a more, normalized database like structure that should support all existing requirements, as well as some future planned functionality.

This document will outline naming conventions and Cloud Code usage.

**Note**: I have not added all the columns to all the tables in Parse. Many (not all) of the fields from the old Parse application created by Pixel will need to be added (with the appropriate naming convention, see below). The UML/Data Diagram **MUST** be updated as the Parse schema changes.

**Data Structure:**

**Terminology**:

In Parse, they don’t have tables, they have Classes. These Classes have rows and columns. They are basically a type of table. Due to this table like nature, I will typically refer to them as tables.

**Relationships**:

The Parse documentation specifies multiple ways to implement one-to-many, many-to-one and many-to-many relationships. Unless there is a specific requirement to use a different choice, we will be exclusively using the “Relationship” method.

<https://parse.com/docs/relations_guide#manytomany-relations>

One-to-one relationships will again, unless a specific requirement prevents us, we will be using the “Pointer” method.

If you encounter a situation that requires the use of a different way of representing a relationship, please consult with Randell.

**Naming**:

Tables will be ProperCased and singular.  
Example: UserLocation (not userLocation, or UserLocations)

Columns will be pascalCased and singular. If the column represents a relationship that refers to multiple items, it will be plural.

Example:  
single value field: name  
multi-value field (using a relationship): locationTriggers

If a column refers to another table (either thru a relationship or pointer) that column’s name should be the name of the table, but pascal cased.

Example 1:  
ReminderGroup table has a column that implements a many type relationship to Reminder. The column name should be reminders.

Example 2:  
The WeatherTrigger table has a column that implements a single type relationship to UserLocation using a pointer. Since this is singular, the column name should be userLocation.

Deviation:  
There will be times when using the pascal cased table name as the column name will not make sense, or will not be clear. In this case, use a unique prefix that describes what you are trying to do and then use the table name (either singular or plural) as a suffix.

Example 1:  
The Reminder table has a reference to itself. This is there to allow a Reminder to be dependent on one or more other Reminders. These Reminders are referred to as parent reminders. Having a field named reminders in the Reminder table wouldn’t make sense. That name would not convey a meaning to the developer. In this case, we use the name parentReminders.

This is an exception and thus should not be needed that often.

**Cloud Code:**

Parse gives us the ability to move business logic into “the cloud”. What many of us would think of as web services, Parse calls Cloud Code.

**Important**: The iOS client needs to be modified so that **ALL** CRUD (create, read, update, delete) operations on Parse data go thru a Cloud Code function. It should **NEVER** query the data directly.

Cloud Code is written in javascript and has some AMD functionality that allows you to include external modules such as \_underscore for extended functionality.

**Naming:**

Follow typical javascript naming patterns, pascal cased function names, etc.

**Note**:

I have not yet completely defined coding standards here. They will evolve as I evaluate how we are using. I will attempt to keep this document updated.

**Important**: Formatting of the javascript is **VERY VERY** important to me. Make you use appropriate white-space as well as tabs for indenting. The code must look neat, easy for a person’s eye to follow and be documented only when you need to explain why you are doing something.

Editing:

I have created and will update a type of parse project. It is configured to push code into the freeminders-rlh project. It uses a stand alone 64 bit version of Sublime as a javascript IDE. You do not have to use sublime to edit the javascript, you can use whatever editor, even notepad if you want. Just make certain the code is formatted properly, including white-space and tabs.

Directory structure:

Parse

/Sublime  
A stand alone x64 installation of sublime and all the supporting modules installed.

parse.exe – The parse deployment tool, a power shell app. When you are ready to deploy changes, in a power shell console, run parse deploy (see parse.com’s windows documentation)

ParseConsole.exe – Used to create a new parse application structure. It should not be needed.

/freeminders/parse

/cloud  
This folder contains all the javascript files that will be pushed to parse.

/config  
This contains the json config object telling the parse deployment tool, parse.exe which application to push the code to

/public  
At this time, I don’t know.

freeminders.sublime-project  
freeminders.sublime-workspace  
These are sublime project files.

parse.js  
This is the parse javascript sdk. Sublime will attempt to use this to provide \*some\* intellisense. You can refer to this file and online documentation on what is available in Parse.