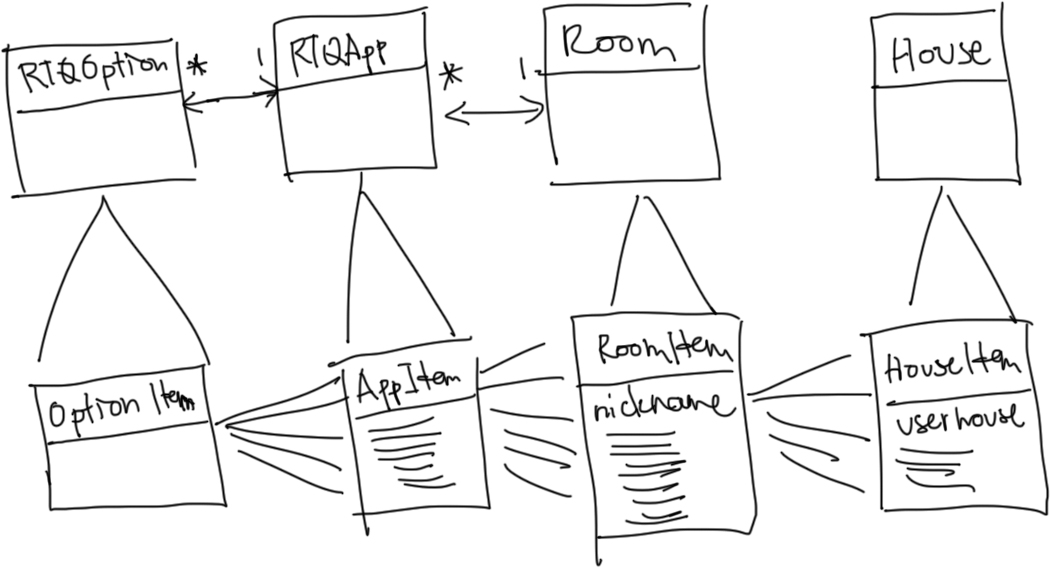
# RTQ Database Documentation

By Radney Aaron Alquiza  
April 29, 2013  
v0.2

## MAIN RTQ PROCESS

Using the "shopping cart" convention, we must create items that will contain the original object properties from the database and add them to their respective containers.  
  
At the end of the line, the **HouseItem** will be contained in the **Quote** object.  
  


## DATABASE SETUP

Basically, we must create 2 sets of tables for the actual RTQ section of the web application: 1 set for the data that the manager enters, and the other for adding into an actual invoice.

This will consume double the data space but will make it easier for managing reusable data through the usage of the application. The problem with previous builds of the RTQ is that the **RTQOptions** themselves are locked into a **Quote** with their foreign key IDs. That shouldn’t happen in a real world application, especially if it is a “shopping cart” style app where lots of similar items will be reused in different carts or in this case, **Quotes.**

Since most of the RTQ is “custom” and there is a new object every time, this is a good idea so that there will be less reference and locking problems even when we delete some items, carts, etc.

The tables will be:

RTQOption -> OptionItem (the actual cleaning service)  
RTQApp -> AppItem (the appliance/furniture) **contains** OptionItems  
Room -> RoomItem ( the room) **contains** AppItems  
HouseItem ( the user’s entered house) **contains** RoomItems

Quote -> **contains** HouseItem

\* APPENDED : THE HOUSE TABLE IS NOT NEEDED.

To copy over the properties from the original objects, when adding an item into a container, use a method to copy over the properties BEFORE adding the item into a container.

Secondary Tables:  
**CleaningType, HouseType, Area** - these tables will just contain a preset (manageable via backend) set of items for a dropdown selection in the UI

In the **Employee**, **type** is a manager, admin or employee

A **PresetItem** is a promotional package. This can be used to directly create a quote, then just use external variables like house area, etc.

# Problems with web development (and how I fixed them):

I’ve come across a bunch of problems (first time I faced these and I’m thankful that I did) while developing this application.

1. Page Updates – since I’m dedicated in delivering as informative and smooth data juggling throughout the website, I’ve used mixed methods from C# ASP.net to jQuery to AJAX. For example, updating data from a table will be very smooth now with the help of jQuery and AJAX calling the server-side methods. I can update data within a table with a slick interface that’s very intuitive and almost automatic. Validation and error handling should be robust and notifications (errors & successes alike) will be unobtrusive and informative.  
     
   - On the spot updates on table data is implemented.  
   - Page update after data update, to show the new data.  
   - Page update after data add, to show the new data. Now I won’t need to manually refresh the page as a normal user just to see new data.  
   - Normally, a developer would use UpdatePanels and other methods to show partial updates but I am using plain jQuery+AJAX alongside C# code to put these all together. Check the code for algorithm comments.

Each HouseItem in a quote has 1 houseType, 1 cleaningType & 1 area.  
A HouseItem is a copy of an Address of a Customer.