TinyDb_export_import

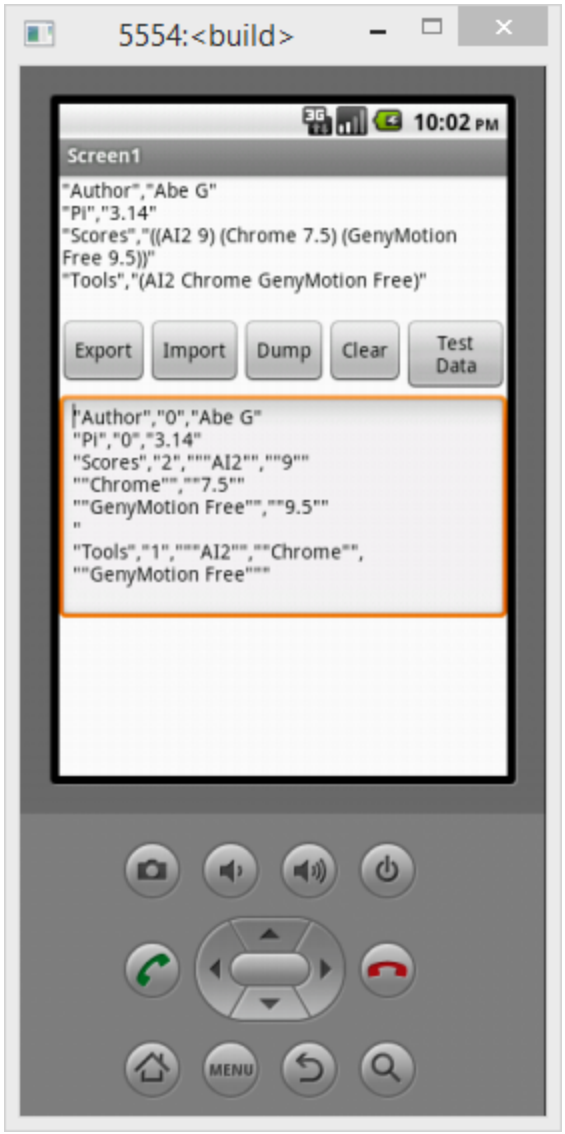
Oct 2018

Background

This app was developed to show how to export and import the complete contents of a TinyDB into and out of a block of text, suitable for writing and reading to and from an external text file for backup and recovery purposes.

This is not meant as an introduction to TinyDB or to list processing. It is meant to be useful in its own right as a tool. It's not necessary to fully understand the internals of a tool to use it, though it helps to understand the limitations of a tool.

Sample Run



This is the screen display of a sample run, using my sample data.

The top section contains a dump of my TinyDB sample, with four tag/value pairs, shown in the lazy and non-reversible AI2 list to text transformation, using parentheses and spaces to wrap list items.

The middle section has some buttons to exercise the various facilities , explained later.

The bottom section has the exported text of the TinyDB contents, complete with tags, values, and an extra dimension column needed to help reconstitute the values when re-importing the text into a new TinyDB.

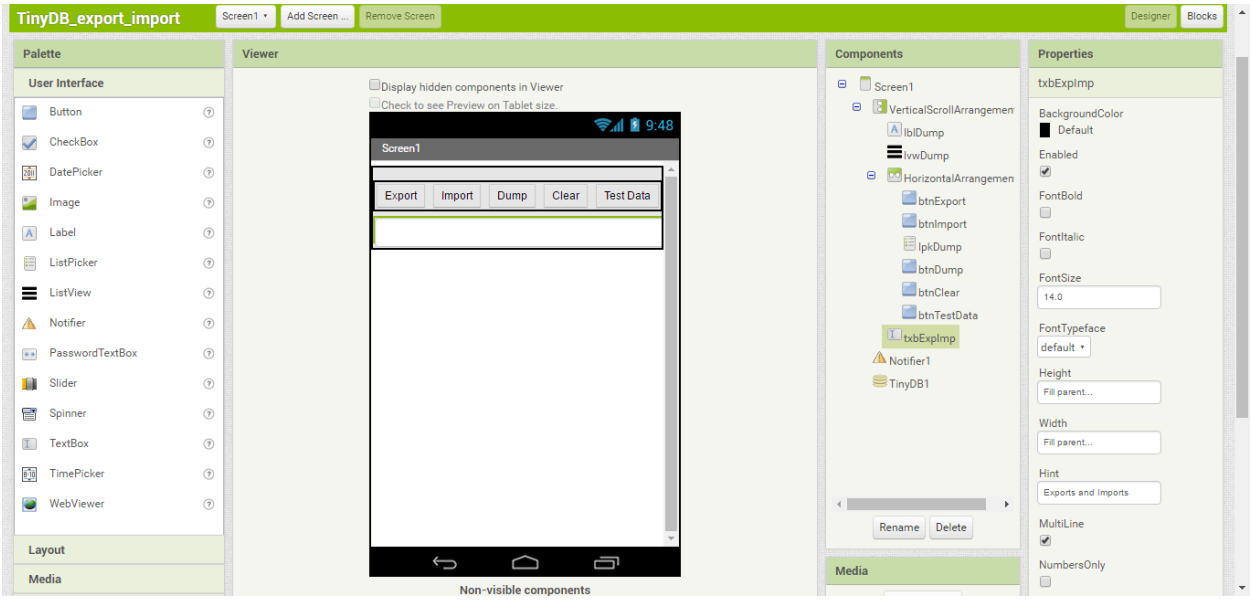
My sample data

I included four pieces of data, to show off how I handle simple values, lists, and tables.

- Simple values (0 dimensions)
    - Tag = “Author”, value = “Abe G”
    - Tag = “Pi”, value = “3.14”
- Lists (1 dimension)
    - Tag = “Tools”,
        - Value =
            - “AI2”
            - “Chrome”
            - “Genymotion Free”
- Tables (2 dimensions)
    - Tag = “Scores”,
        - Value =
            - “Ai2”, 9
            - “Chrome”, 7.5
            - “Genymotion Free”, 9.5

I have listed my sample data here in ascending order by dimension, from simplest structure (plain values) to complex (two dimensional tables.) My sample data displayed by the app comes out in tag order, which TinyDB returns alphabetically by tag.

Designer



Components

lblDump

A label for output of a dump of TinyDb. A dump is a haphazard lazy export, too jumbled for re-import.

lvwDump

A ListView for output of a dump of TinyDb. A dump is a haphazard lazy export, too jumbled for re-import.

btnExport

The Export button. It formats TinyDB into one piece of text, then sends it to the output text box.

btnImport

The Import button. It takes the text from the output text box, reforms it into its original tags and values, and loads them into TinyDB.

btnDump

The Dump button. It sends the contents of TinyDB to lblDump.

btnClear

The Clear button. It clears TinyDB.

btnTestData

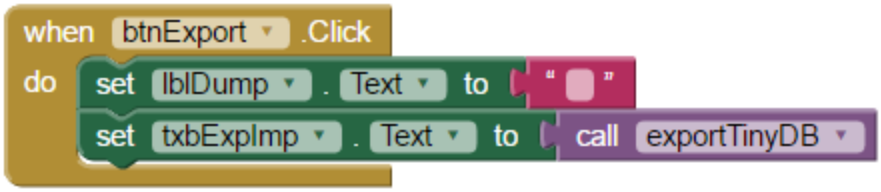
The Test Data button. It loads test data into TinyDB.

txbExpImp

The Export/Import Text box. Target for export, source for Import. A Text Box is better for this than a Label, because you can cut and paste it in an emulator.

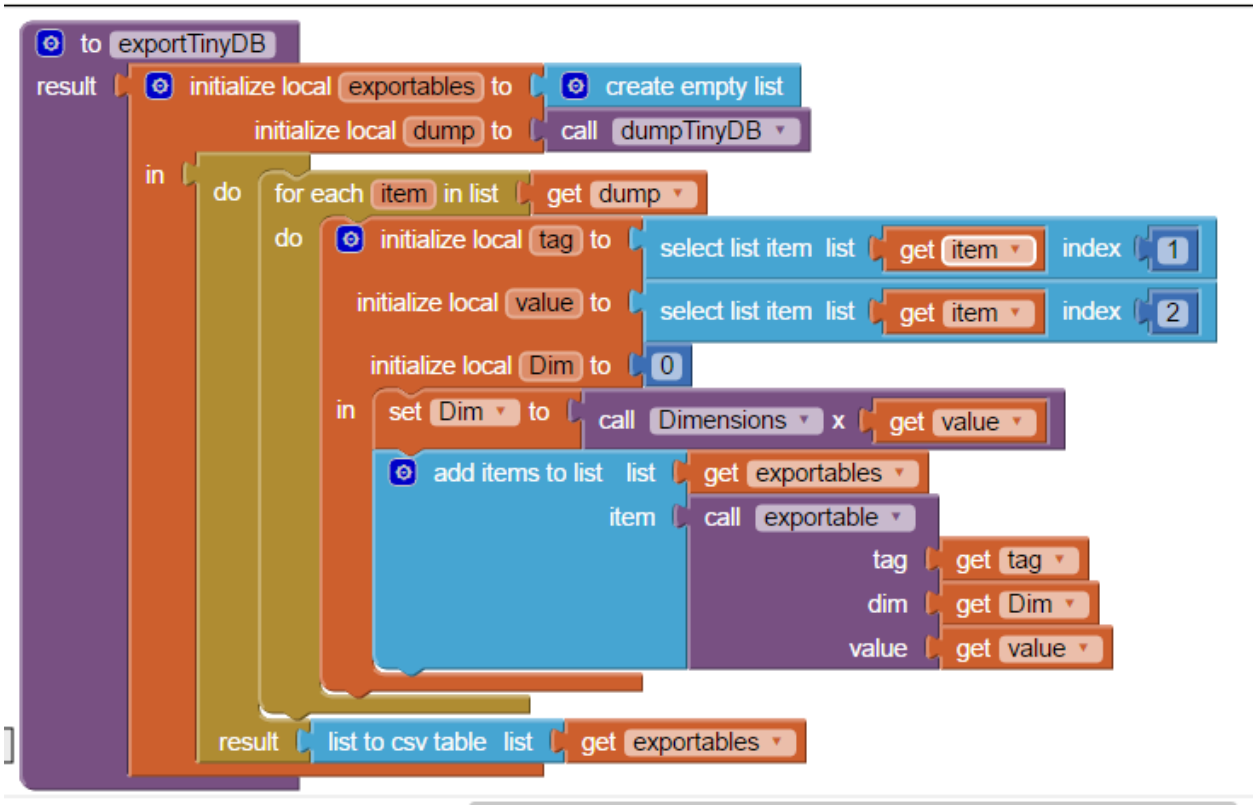
Blocks

When btnExport.Click



The Export button clears label lblDump to make room on the screen for txbExpImp, which is loaded from procedure exportTinyDB.

exportTinyDB



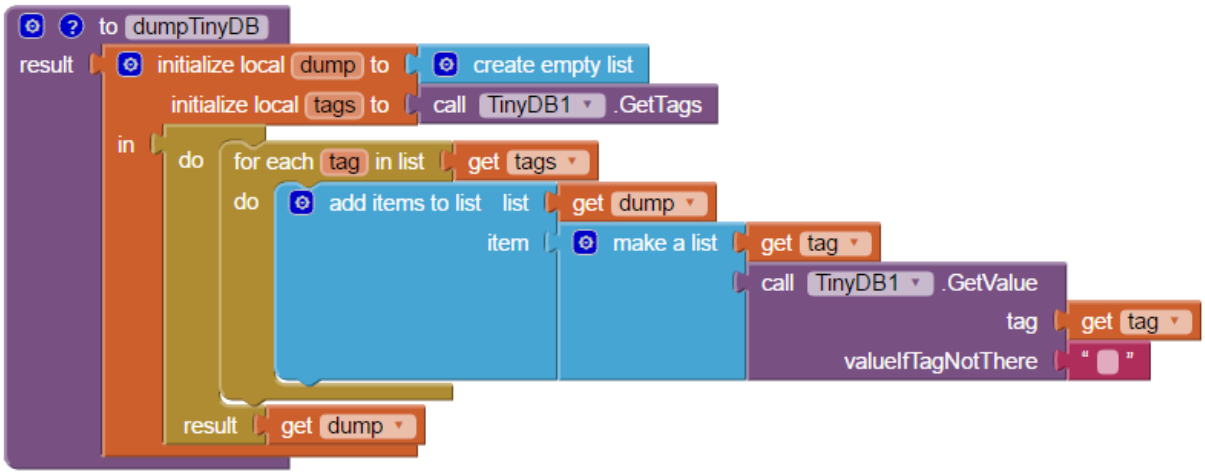
This is a value procedure, returning a Comma Separated Values (CSV) text representation of a table of the exportable representation of each (tag,dimension,value) row.

It works in three phases:

1. Extract all (tag/value) pairs from TinyDB into local variable **dump** using procedure dumpTinyDB
2. For each dumped pair (tag/value), get the dimension (0/1/2) of that value
3. Build an exportable row suitable for adding to a 2 dimensional table representing the tag, dimension, value

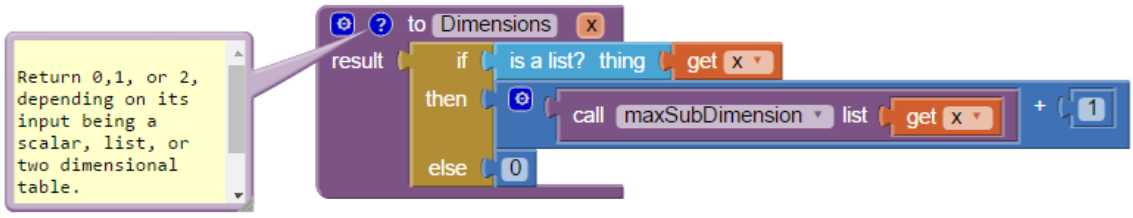
Finally, it returns the CSV Table text.

dumpTinyDB



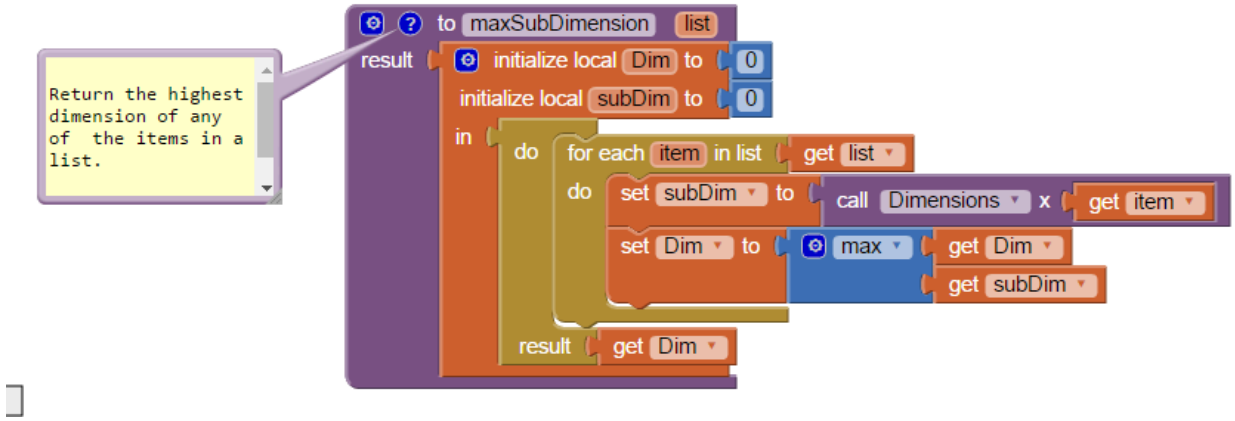
This value procedure returns a two column table with each row containing a TinyDB tag in column 1, and its corresponding TinyDB value in column 2. Notice that this works only in Tinydb1. If you want to use multiple TinyDB instances with different name spaces, you will have to add that yourself.

Dimensions



To determine the number of dimensions of a structure, we divide up the question recursively using a helper procedure maxSubDimension. If the structure is not a list, it has dimension 0. Otherwise, it's at least a list and its dimension is 1 more than the highest dimension of the items in that list.

maxSubDimension

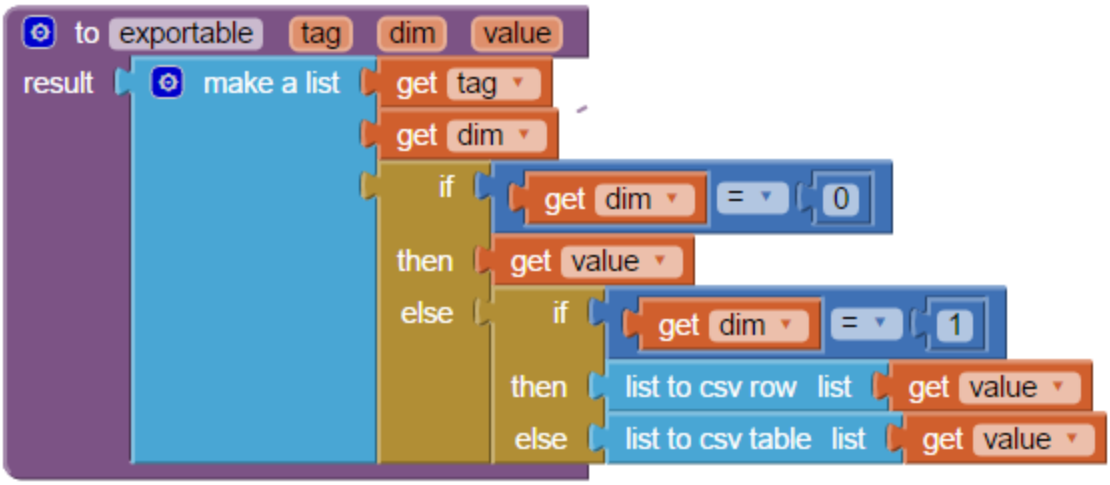


This can get a bit mind boggling. We are about to use recursion, where a procedure calls another procedure that may have called it. The recursion must eventually end, as long as it is working with a value from TinyDB, which filters out abnormal lists like the notorious Ouroborous.

We return a local variable Dim, which is initially 0, but is compared to the dimension of each item in the input list, and is set to the maximum of itself or that new dimension value.

After exiting the for each loop, variable Dim will have the maximum value of the dimensions of all the items in the given list.

Exportable

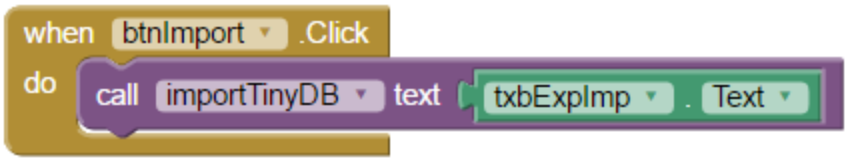


To make an exportable representation of a TinyDB tag and its value, we have to flatten the value so it will fit into a piece of text in the 3rd position of a list (tag, dimension, flattened value). Fortunately, we have available the dimension (0/1/2) of the value in our second input parameter.

A 0 dimensional value is okay in itself, and can be used directly. A 1 dimensional value can be run through the **list to csv row** block to turn into a CSV row text. A 2 dimensional value can be run through the **list to csv table** block to turn into a CSV table text. If the dimension is higher than 2, all bets are off, and the app developer should read up on the relational model then redesign his data structures.

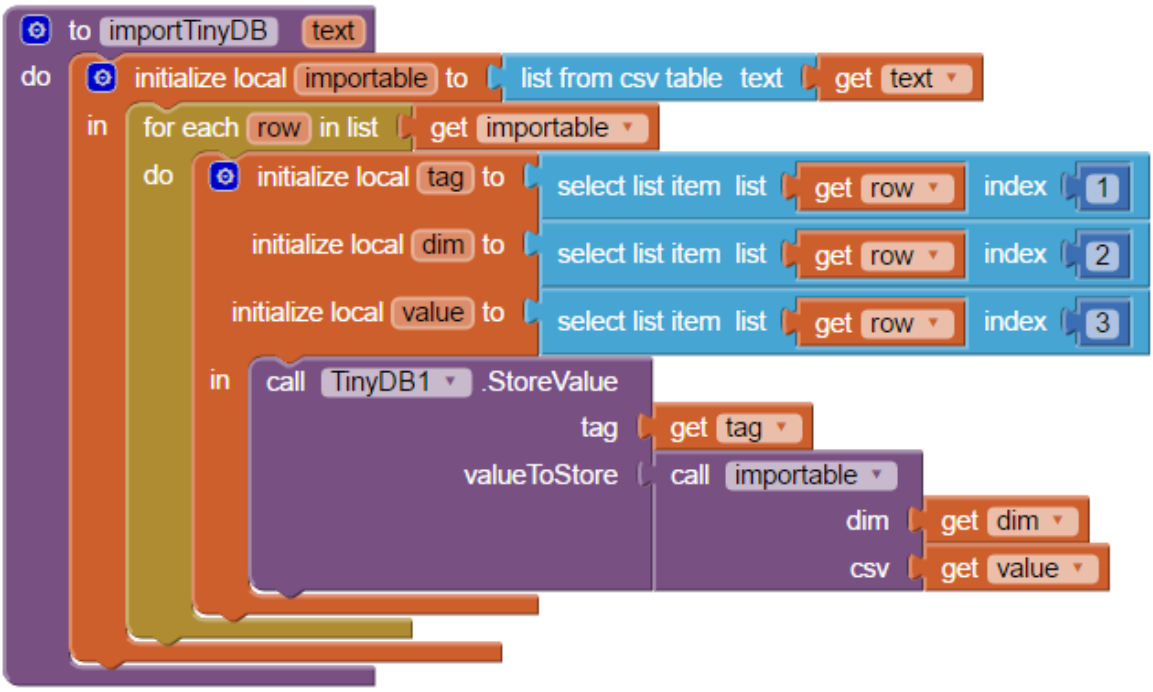
The output of this procedure will be a 3 item list (tag, dimension, flat text value).

When btnImport.Click



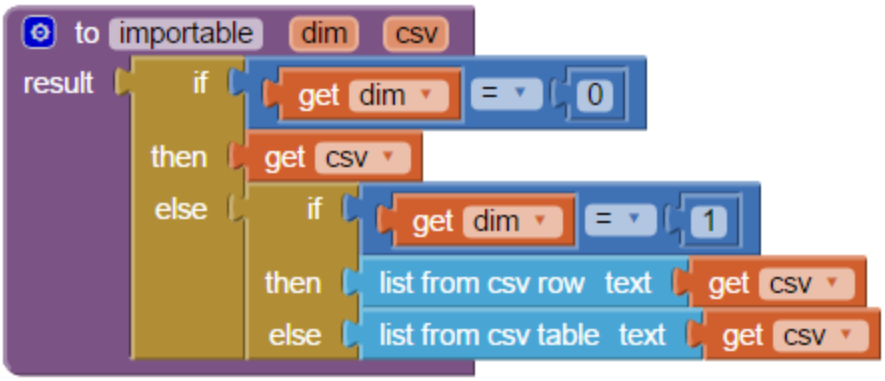
The Import button takes the contents of text box txbExpImp and loads it into TinyDB1 through procedure importTinyDB.

importTinyDB



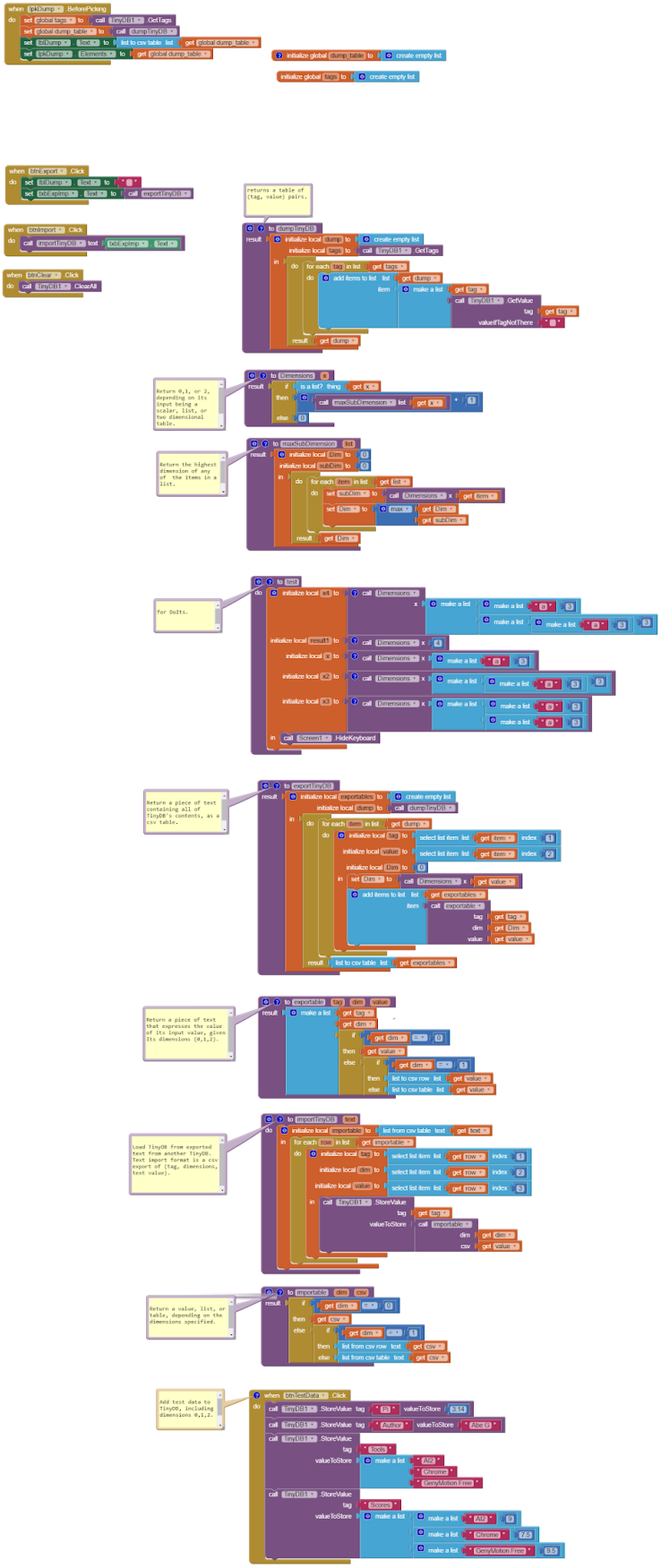
This procedure does the opposite of the exportTinyDB procedure. It accepts a CSV Table text block and runs it through a list from csv table block to get a three column (tag,dimension, exported value) table.
For each row in that table, that row is transformed back into its original tag/value pair using the importable procedure and the given dimension (0/1/2) number of that value, then stored into TinyDB1.

Importable



This value procedure does the opposite of the exportable procedure, reconstituting a simple value, list, or table from a text value and its dimension number, using the **list from csv row** or **list from csv table** blocks, as appropriate.

All blocks (in case I missed something)



Gallery Link

ai2.appinventor.mit.edu/?galleryId=6019316599488512

Other Projects

https://docs.google.com/document/d/1acg2M5KdunKjJgM3Rxpy_Rf6vT6OozxdIWglgbmzroA/edit?usp=sharing