



jagga
CLIMATE
DESIGNERS

#OurGiantsAreFemale

Margaret Ingels

- October 25, 1892 – December 13, 1971
- First female engineering graduate from the University of Kentucky
- Joined the American Society of Heating and Ventilating Engineers Research Lab
- Developed the “effective temperature” scale to incorporate humidity and air movement in the equation for human comfort
- “Petticoats and Slide Rules” is a famous speech and inspiration to women in engineering



A Tree Full of Object Data

**Decomposing object data into a tree structure
using JSON and SQLite**

Agenda

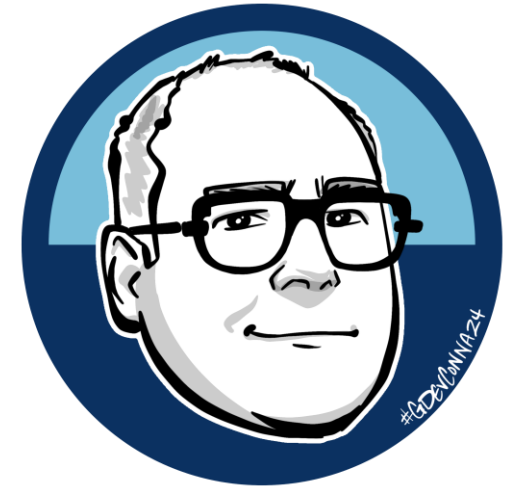
- **About Me**
- **Warning**
- **Problem Statement**
- **Solution**
 - **Object to JSON**
 - **JSON to tree**
- **Use Cases**
- **Bonus**
- **Links**



About Me

- **Stefan Lemmens**

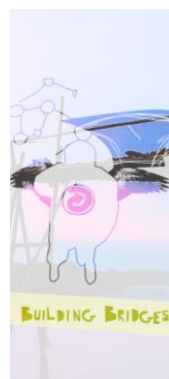
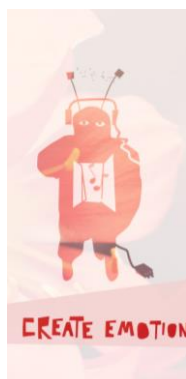
- Test development engineer at Jaga Climate Designers
- LabVIEW Developer since 1999 (LabVIEW 5.0)
- LabVIEW Champion since 2021
- Member of SAS LabVIEW Mastermind
- Created a leaderboard on Advent of Code and Discord server for LabVIEW developers
- [LabVIEW shortcuts booklet](#)
- Mostly use Actor Framework ([basics to PPL tutorial](#))
- Mail : stefanlemmens.be@gmail.com
- NI Forum : StefanLemmens
- Twitter : @stefan_Lemmens





CLIMATE
DESIGNERS

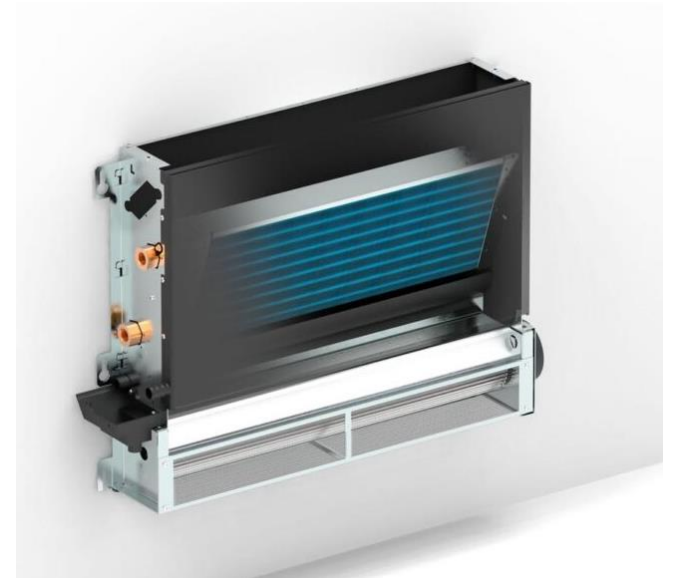
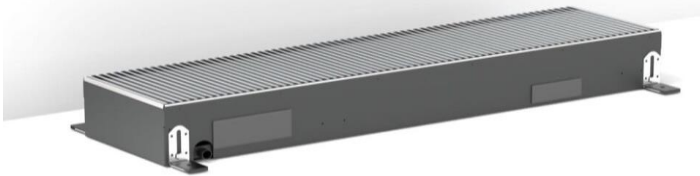
**Ensuring a better indoor climate without
damaging the outdoor climate.
That's our main goal as Jaga Climate Designers!**



jaga CLIMATE DESIGNERS



Jaga Products



Warning

- This works for me
- Still room for improvements
- This will break encapsulation
 - Use it carefully

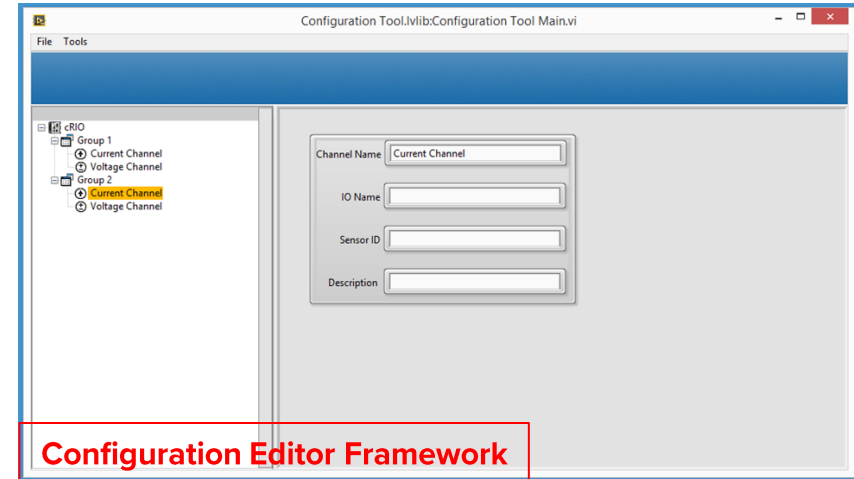
Warning

- This works for me
- Still room for improvements
- This will break encapsulation
 - Use it carefully

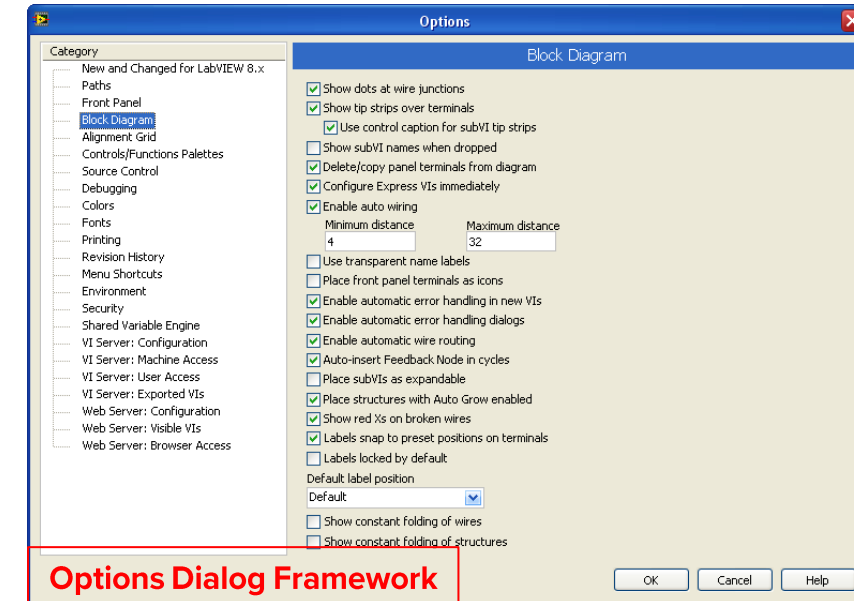
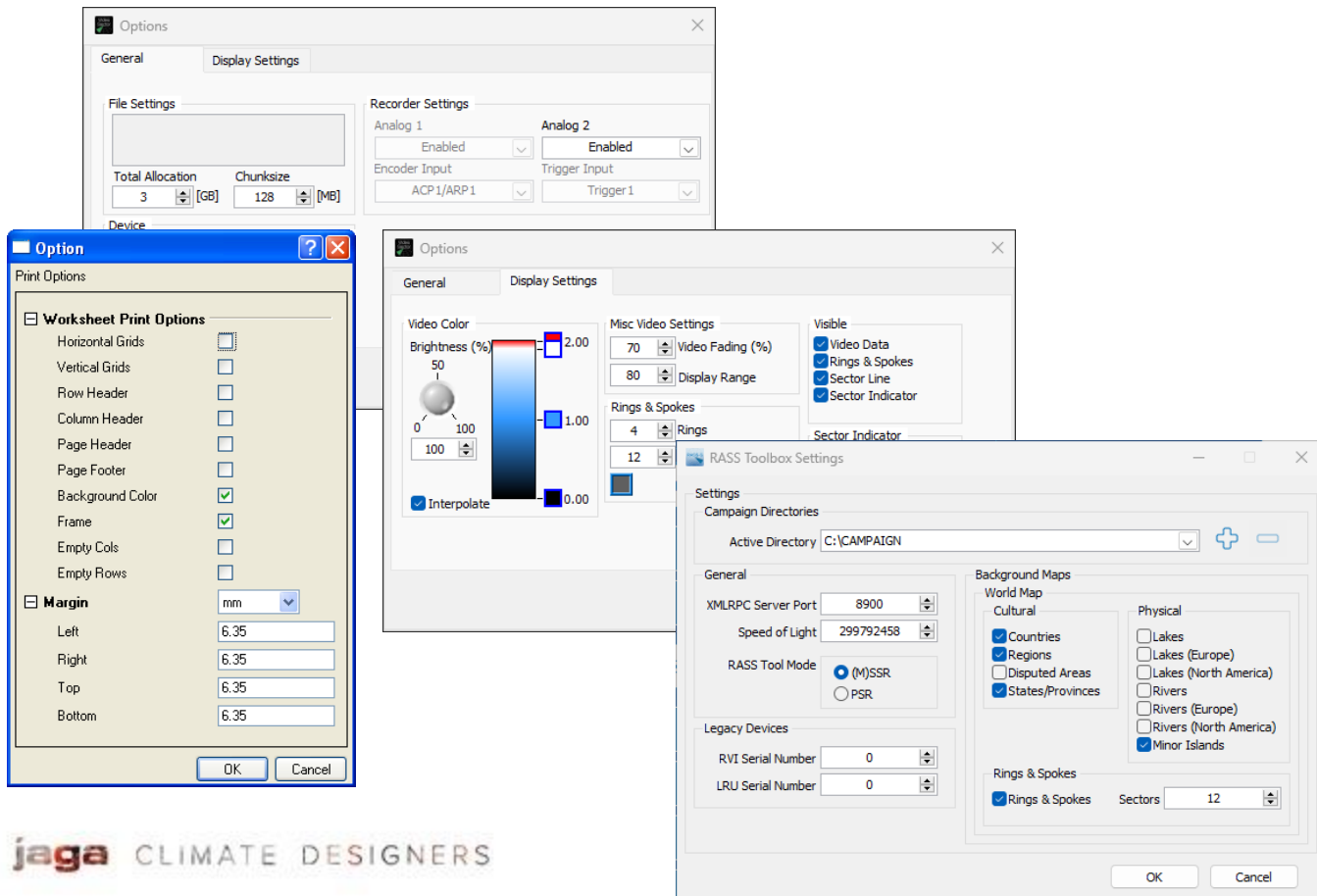


Problem Statement

- Different applications/test have different settings
- Configuration clusters/objects can contain a lot of parameters
- A different UI is needed each time



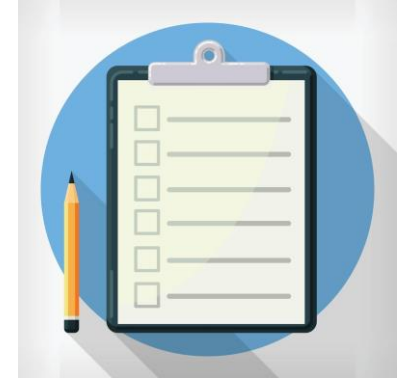
Configuration Editor Framework



Options Dialog Framework

Requirements

- 1 **Generic UI**
- User-friendly
- Take a (configuration) object and
 - ✓ Display
 - ✓ Edit
 - ✓ Read from file
 - ✓ Save to file



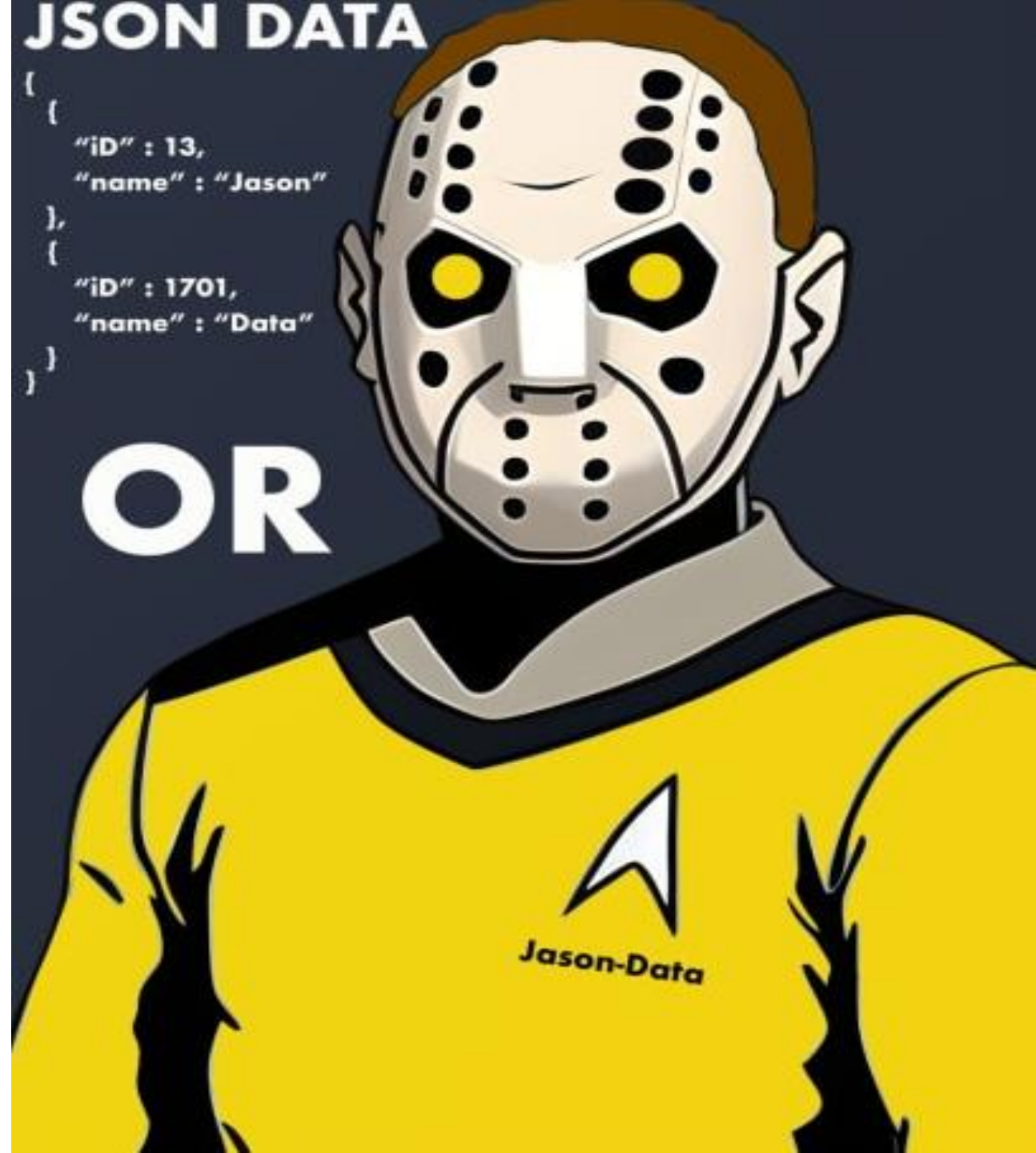
Solution

- Part 1 : Object to/from text/file
- Part 2 : Display & edit the data

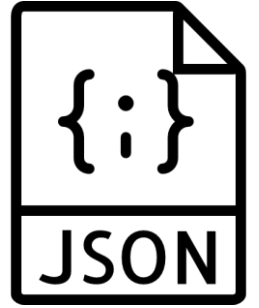


File Format

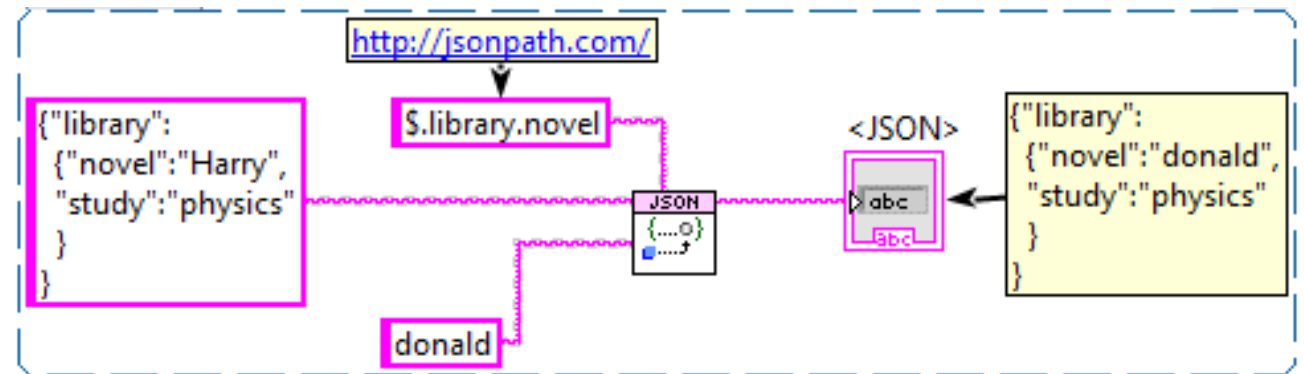
- I wanted to switch from XML to JSON
- Why????
 - ✓ Easier to read and parse
 - ✓ Better support for (empty) arrays
 - ✓ Supports optional elements easier
 - ✓ Way to represent objects
 - ✓ Better suited for data storage
 - ✓ Used by Jaga Device Controllers



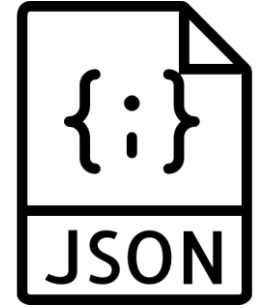
JSONtext



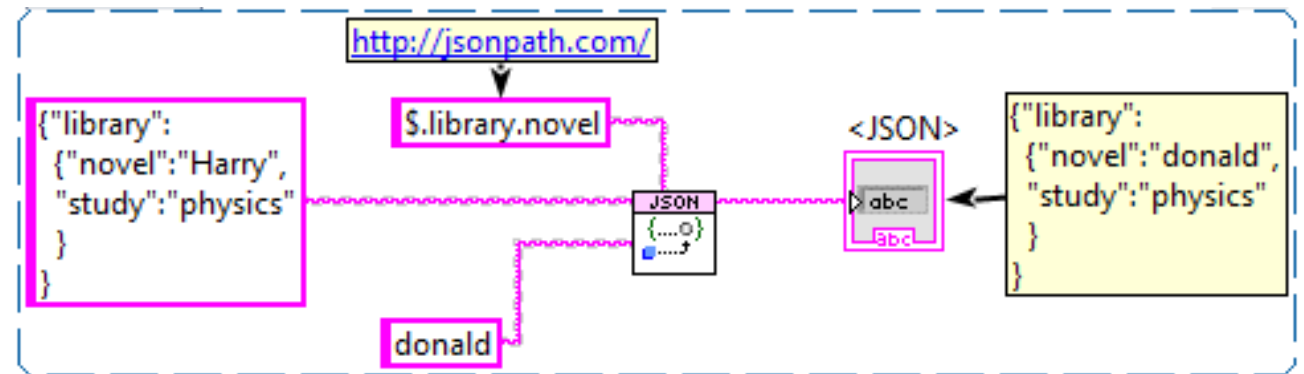
- JSONtext by JDP
 - ✓ Uses JSON Path notation to access (sub)items
 - ✓ Very good for everything with JSON
 - ✓ Now also support JSON scheme validation



JSONtext

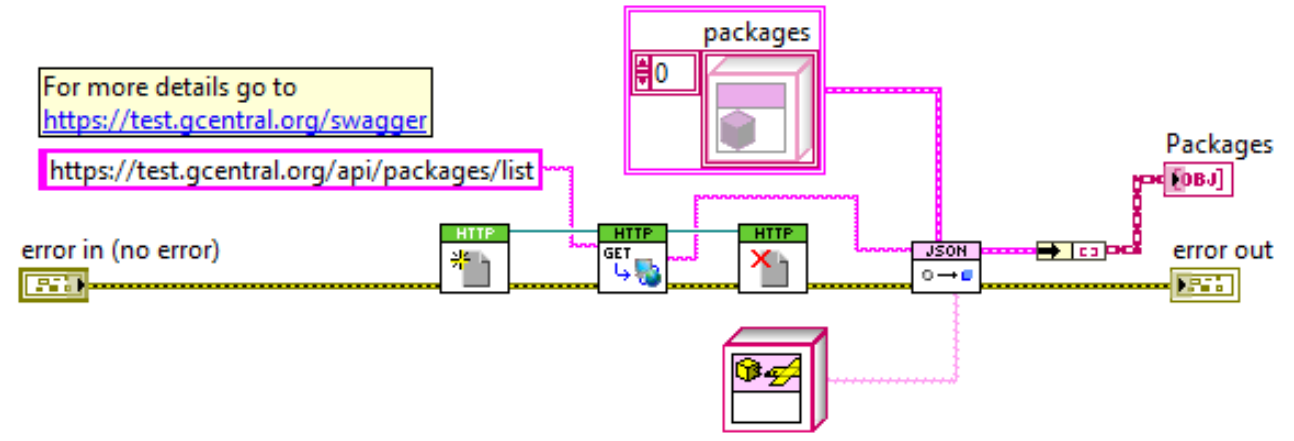
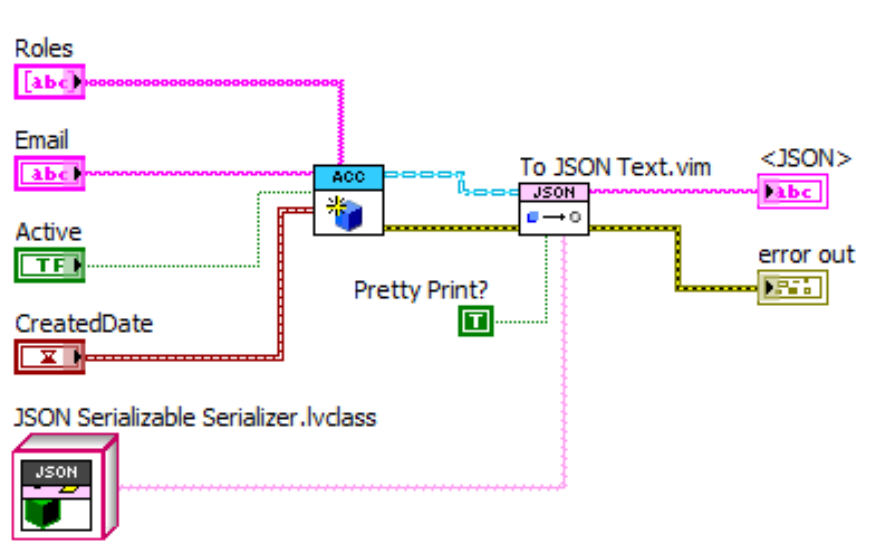
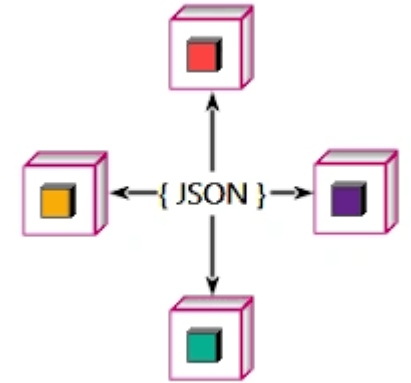


- JSONtext by JDP
 - ✓ Uses JSON Path notation to access (sub)items
 - ✓ Very good for everything with JSON **but doesn't support objects**
 - ✓ Now also support JSON scheme validation



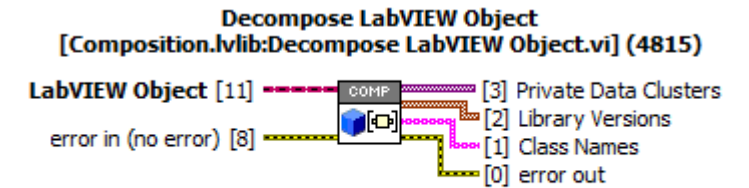
JSONtext Object Serialization

- VIPM toolkit by PNR
- Extension for JSONtext to (de)serialize objects
- Easy to use
- **Breaks encapsulation !!!!**

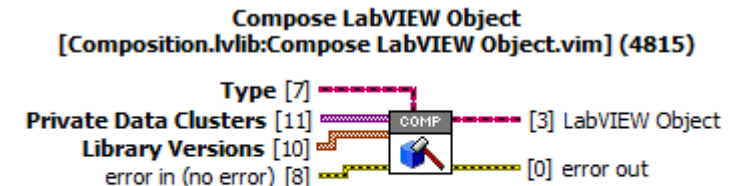


LabVIEW Composition

- Another VIPM toolkit by PNR
- Provides functions to compose and decompose
 - objects
 - clusters
 - maps and sets
- Breaks encapsulation
- Makes private data accessible
- Easy to use
- Used by JSONText object serialization



Decomposes a LabVIEW Object into its private data clusters from oldest ancestor to last descendant (same order as in Probe Watch Window).



Composes a LabVIEW Object from its private data clusters and specified library versions.

Displaying the data

- Generic way to display all kinds of data in a user-friendly way

Tree Structure

- Can handle multiple levels
- Can handle different data types
- Commonly used to display JSON

Custom Properties

Name	Value
Name	New
Wait	0
Settings	
temperatures	
RoomTemperature	25
WaterTemperature	0
WaterTemperature2	0
jdpc_config	
controlType	
ACParams	
TPTparams	
mode	
desiredTempCelciusHeating	0
desiredTempCelciusCooling	0
tempDiffCelciusHeating	0
tempDiffCelciusCooling	0
pipeHeatingMinTempCelcius	0
pipeCoolingMaxTempCelcius	0
tempDiffCelciusTurnOffHeating	0
tempDiffCelciusTurnOffCooling	0
tempDiffCelciusHysteresisHeating	0
tempDiffCelciusHysteresisCooling	0
fanSpeedRegRangeTempCelciusHeatin	0
fanSpeedRegRangeTempCelciusCoolin	0
speedConfigurationsPercent	
H	
C	
BMSparams	
commonParams	
fanConfiguration	
type	
minSpeed	0
maxSpeed	0
3KeyControlPanelConfiguration	
centerDesiredTempCelciusTPT	
3KeyControlpanelPresent	false
CO2SensorPresent	false
valveConfiguration	0

Show JSON in a tree

- Let's ask ChatGPT

Show JSON in a tree

- Let's ask ChatGPT



Tree Control - A little present

- From Steve Watts (Random Ramblings on LabVIEW Design blog)
- Create a tree hierarchy from a database table

The screenshot shows a LabVIEW front panel titled "TreeTestStubExperiment.vi Front Panel". It features a data flow diagram with the following components:

- dBTableToTree**: A table control with columns "Id(PK)", "parent", "name", and "tag". It contains 9 rows of data.
- Tree**: A tree control displaying a hierarchical structure based on the table data.
- TreeTodBTable**: A table control with 6 rows, representing a processed version of the data.
- INSERTSQLString**: A text area control for generating SQL insert statements.
- error in (no error)** and **error out**: Error handling controls with status and code indicators.

The data in the **dBTableToTree** table is as follows:

Id(PK)	parent	name	tag
1	0	Category 1	
2	1	Menu Item X	
3	1	Menu Item Y	
4	1	Menu Item Z	
5	0	Category 2	
6	5	Other Menu 1	
7	5	Other Menu 2	
8	7	Menu Lvl3	
9	0	Empty Category	

The **Tree** control displays the following hierarchy:

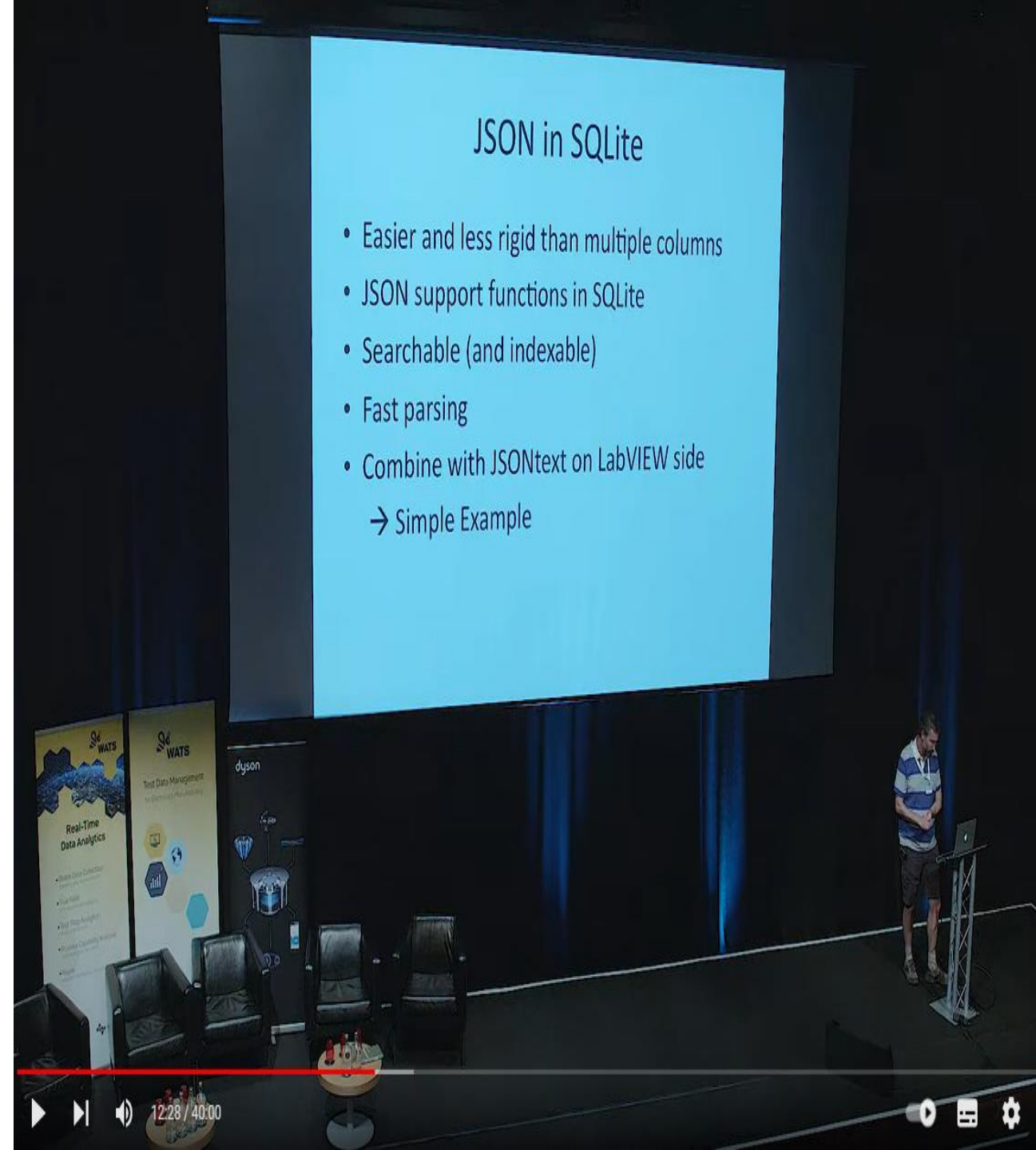
- Category 1
 - Menu Item X
 - Menu Item Y
 - Menu Item Z
- Category 2
 - Other Menu 1
 - Other Menu 2
 - Menu Lvl3
 - Empty Category

The **TreeTodBTable** table contains the following data:

Id	parent	name
1	0	Category 1
2	1	Menu Item X
3	1	Menu Item Y
4	1	Menu Item Z
5	0	Category 2
6	5	Other Menu 1

JSON to database table

- Combine JSONtext and SQLite to create the database table
- Presented at GDevCon#2 by Dr. James Powell

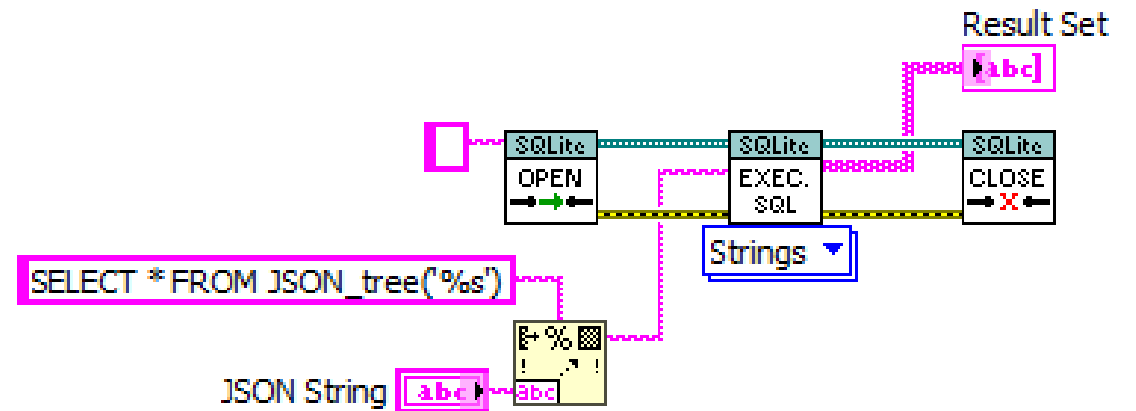


Example data

```
1  {
2  "Actors": [{
3      "name": "Tom Cruise",
4      "age": 56,
5      "Born At": "Syracuse, NY",
6      "Birthdate": "July 3, 1962",
7      "photo": "https://jsonformatter.org/img/tom-cruise.jpg",
8      "wife": null,
9      "weight": 67.5,
10     "hasChildren": true,
11     "hasGreyHair": false,
12     "children": [
13         "Suri",
14         "Isabella Jane",
15         "Connor"
16     ]
17 },
18 {
19     "name": "Robert Downey Jr.",
20     "age": 53,
21     "Born At": "New York City, NY",
22     "Birthdate": "April 4, 1965",
23     "photo": "https://jsonformatter.org/img/Robert-Downey-Jr.jpg",
24     "wife": "Susan Downey",
25     "weight": 77.1,
26     "hasChildren": true,
27     "hasGreyHair": false,
28     "children": [
29         "Indio Falconer",
30         "Avri Roel",
31         "Exton Elias"
32     ]
33 }
34 ]
35 }
```

JSON_tree

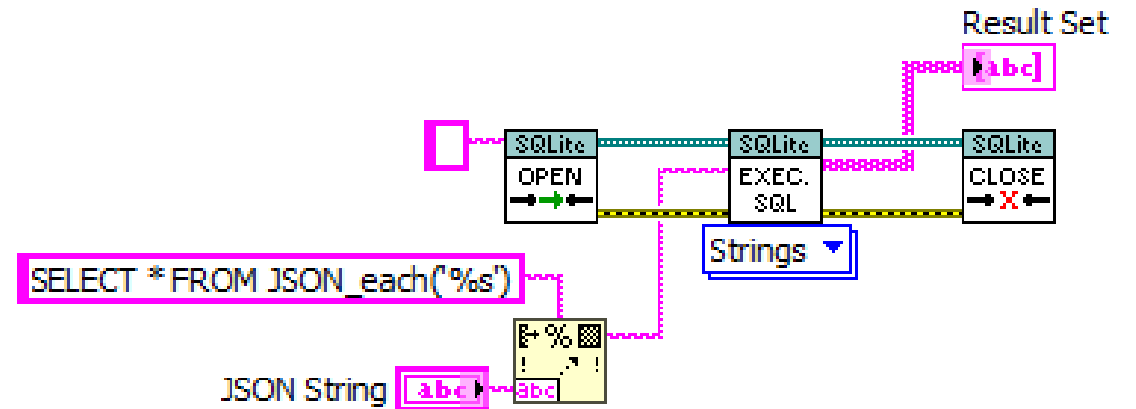
- Can query JSON data
- Returns table with one row for each element
- Walks recursively through JSON substructures



	key	value	type	atom	id	parent	fullkey	path
1	NULL	{"Actors":[{"name":"Tom Cruise","age":56,"Born...	object	NULL	0	NULL	\$	\$
2	Actors	[{"name":"Tom Cruise","age":56,"Born ...	array	NULL	2	0	\$.Actors	\$
3	0	{"name":"Tom Cruise","age":56,"Born ...	object	NULL	3	2	\$.Actors[0]	\$.Actors
4	name	Tom Cruise	text	Tom Cruise	5	3	\$.Actors[0].name	\$.Actors[0]
5	age	56	integer	56	7	3	\$.Actors[0].age	\$.Actors[0]
6	Born At	Syracuse, NY	text	Syracuse, NY	9	3	\$.Actors[0].Born At	\$.Actors[0]
7	Birthdate	July 3, 1962	text	July 3, 1962	11	3	\$.Actors[0].Birthdate	\$.Actors[0]
8	photo	https://jsonformatter.org/img/tom-cruise.jpg	text	https://jsonformatter.org/img/tom-cruise.jpg	13	3	\$.Actors[0].photo	\$.Actors[0]
9	wife	NULL	null	NULL	15	3	\$.Actors[0].wife	\$.Actors[0]
10	weight	67.5	real	67.5	17	3	\$.Actors[0].weight	\$.Actors[0]
11	hasChildren	1	true	1	19	3	\$.Actors[0].hasChildren	\$.Actors[0]
12	hasGreyHair	0	false	0	21	3	\$.Actors[0].hasGreyHair	\$.Actors[0]
13	children	["Suri","Isabella Jane","Connor"]	array	NULL	23	3	\$.Actors[0].children	\$.Actors[0]
14	0	Suri	text	Suri	24	23	\$.Actors[0].children[0]	\$.Actors[0].children

JSON_each

- Can query JSON data
- Returns table with one row for each element
- Only walks one level



	key	value	type	atom	id	parent	fullkey	path
1	name	Tom Cruise	text	Tom Cruise	2	<i>NULL</i>	\$.name	\$
2	age	56	integer	56	4	<i>NULL</i>	\$.age	\$
3	Born At	Syracuse, NY	text	Syracuse, NY	6	<i>NULL</i>	\$.Born At	\$
4	Birthdate	July 3, 1962	text	July 3, 1962	8	<i>NULL</i>	\$.Birthdate	\$
5	photo	https://jsonformatter.org/img/tom-cruise.jpg	text	https://jsonformatter.org/img/tom-cruise.jpg	10	<i>NULL</i>	\$.photo	\$
6	wife	<i>NULL</i>	null	<i>NULL</i>	12	<i>NULL</i>	\$.wife	\$
7	weight	67.5	real	67.5	14	<i>NULL</i>	\$.weight	\$
8	hasChildren	1	true	1	16	<i>NULL</i>	\$.hasChildren	\$
9	hasGreyHair	0	false	0	18	<i>NULL</i>	\$.hasGreyHair	\$
10	children	["Suri","Isabella Jane","Connor"]	array	<i>NULL</i>	20	<i>NULL</i>	\$.children	\$

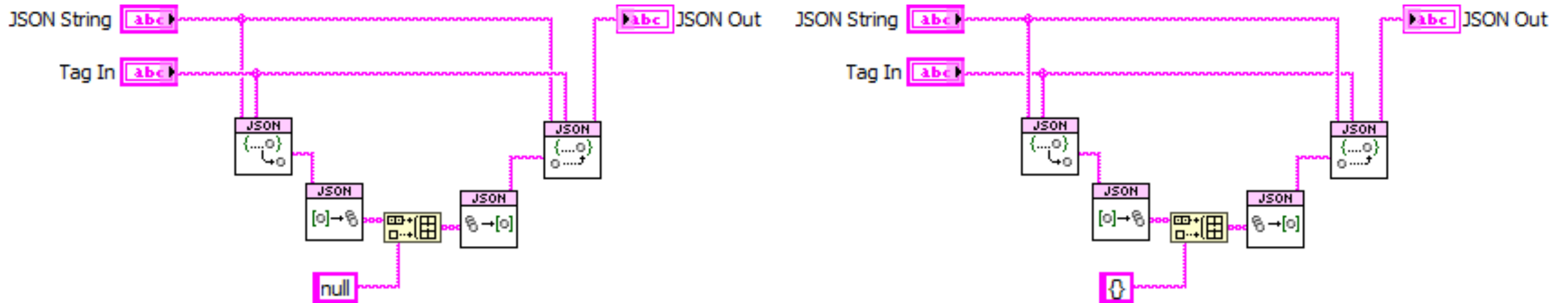
JSON_extract

- Extracts and returns one or more values
- Comparable to unbundle for clusters
- Comparable to using the JSONtext Find Item
- Useful in queries in combination met WHERE-statement

```
SELECT value FROM JSON_each(json('MY_JSON_STRING'), '$.Actors')  
WHERE JSON_extract( value, '$.age' ) > 55;
```

Add empty elements

- Use JSONtext to add a default element to an array
- Without knowing the datatype
- Might give error 402844 : JSONtext: Input JSON cannot be converted to requested LabVIEW Data Type



Use Case 1

- JSON Viewer

The screenshot shows a software application window titled "A Tree Full of Object Data.vi". The window has a menu bar with "File", "Edit", "View", "Project", "Operate", "Tools", "Window", and "Help". Below the menu bar are three icons: a yellow lightbulb, a blue folder, and a blue folder with a document. The address bar shows "LVClass" and the file path "D:\Project...\Test Classes\Test Class Hollywood.lvclass".

The main area is titled "Config" and contains a tree view of a JSON object. The root node is "\$". Under "\$" is a node "Actors". "Actors" has two child nodes: "Actors[0]" and "Actors[1]". Each of these nodes has a list of properties: "name", "age", "Born At", "Birthdate", "photo", "wife", "weight", "hasChildren", "hasGreyHair", and "children".

Below the tree view is a table with two columns. The left column is labeled "Element" and the right column is labeled "Value". The table contains the following data:

Element	Value
\$.Actors	
Actors[0].name	
Actors[0].age	0
Actors[0].Born At	
Actors[0].Birthdate	
Actors[0].photo	
Actors[0].wife	
Actors[0].weight	0
Actors[0].hasChildren	false
Actors[0].hasGreyHair	false
Actors[0].children	
Actors[1].name	
Actors[1].age	0
Actors[1].Born At	
Actors[1].Birthdate	
Actors[1].photo	
Actors[1].wife	
Actors[1].weight	0
Actors[1].hasChildren	false
Actors[1].hasGreyHair	false
Actors[1].children	

At the bottom of the window, there are two input fields. The first is labeled "Element" and contains the text "\$.Actors". The second is labeled "Filters" and is currently empty.

Use Case 2

- Result Screen

```
"Air Temperature":{  
  "Measured":24.472225189209,  
  "Min":24,  
  "Max":26,  
  "Status":"Pass"  
}
```

	Minimum	Measured	Maximum
\$			
Serial Number		0020240102074211	
MAC Address		08:d1:f9:9f:51:bc	
Pass_Fail		true	
Date		20240701	
Time		090809	
Air Temperature	24	24.472225189209	26
Water Temperature	24	24.5252571105957	26
Room Temperature	24	24.5252571105957	26
CO2 Sensor Type		CM1106	
Current In			
Current In 1			
Current In 2			
Vout	0	1.424	10
adcInput1mV	4000	5032	6000
adcInput2mV	4000	5030	6000

Use Case 3

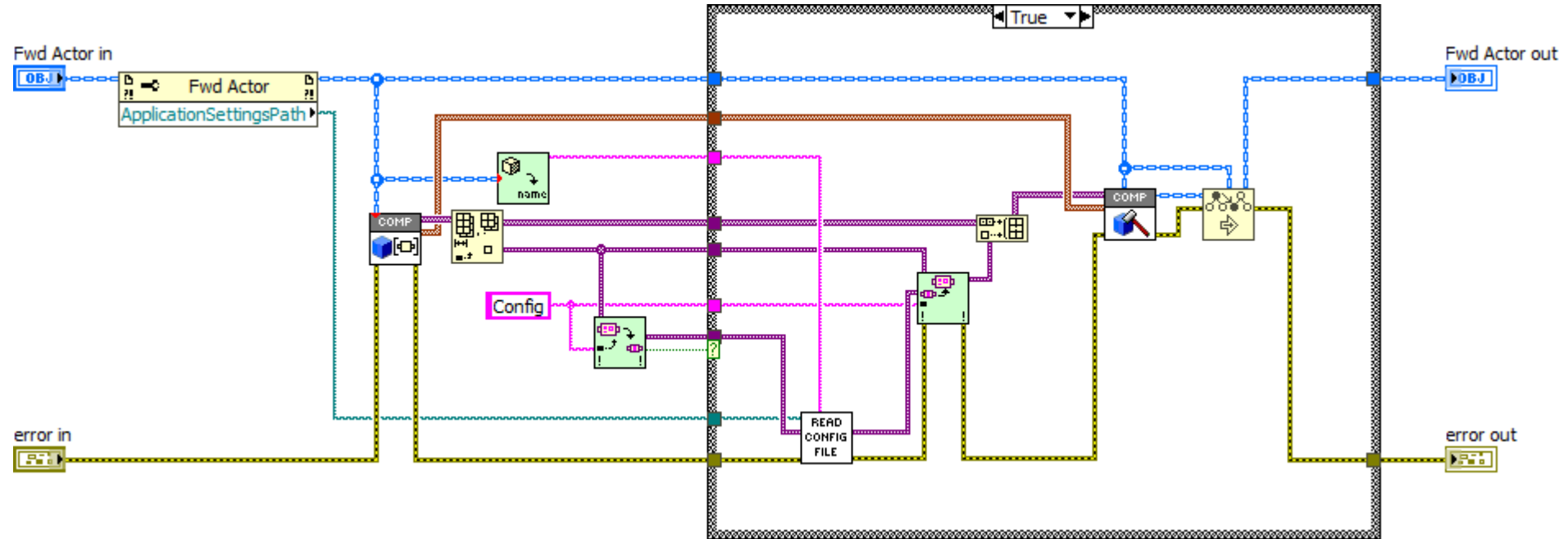
- Config UI

The screenshot shows a 'Settings' application window. On the left is a 'Sections' list with 'Measure' selected. The main area displays a tree view of settings categories and a table of their values.

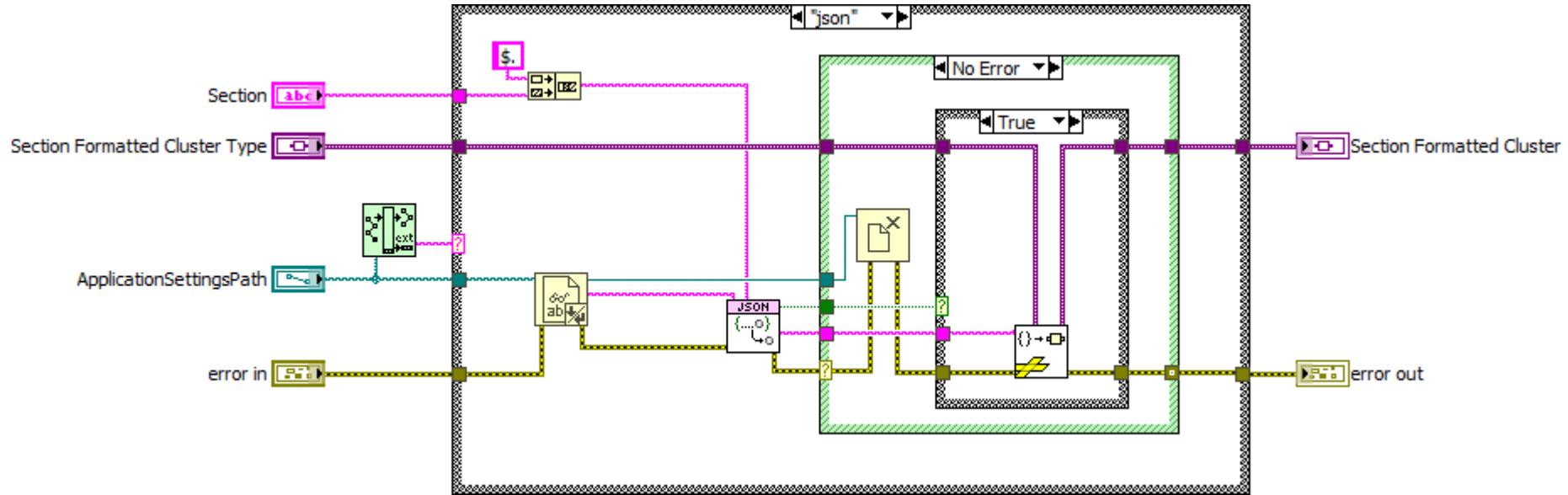
Parameter	Value
Measure	
Settings	
VISA Power Supply	
VISA Analog Input	LC-AI8-01
Ch1 Voltage	12
Ch2 Voltage	5
Ch3 Voltage	5
WiFi_Wait	90000
Limits	
Air Temperature	
Water Temperature	
Min	24
Max	26
Room Temperature	
Min	24
Max	26
Power Supply Ch1	
Current	
Power Supply Ch2	
Current	
Power Supply Ch3	
Analog Input 1	
CO2 Sensor Type	CM1106
adcInput1mV	
Min	4000
Max	6000
adcInput2mV	
Min	4000
Max	6000
Original WiFi Network	IoT

The application footer includes the 'jaga' logo and 'OK' and 'Cancel' buttons.

Base Actor



Read Config from File



Bonus - The other way around

- Creating a LabVIEW cluster typedef from a JSON file

```
1  {
2  "Actors": [{
3      "name": "Tom Cruise",
4      "age": 56,
5      "Born At": "Syracuse, NY",
6      "Birthdate": "July 3, 1962",
7      "photo": "https://jsonformatter.org/img/tom-cruise.jpg",
8      "wife": null,
9      "weight": 67.5,
10     "hasChildren": true,
11     "hasGreyHair": false,
12     "children": [
13         "Suri",
14         "Isabella Jane",
15         "Connor"
16     ]
17 },
18 {
19     "name": "Robert Downey Jr.",
20     "age": 53,
21     "Born At": "New York City, NY",
22     "Birthdate": "April 4, 1965",
23     "photo": "https://jsonformatter.org/img/Robert-Downey-Jr.jpg",
24     "wife": "Susan Downey",
25     "weight": 77.1,
26     "hasChildren": true,
27     "hasGreyHair": false,
28     "children": [
29         "Indio Falconer",
30         "Avri Roel",
31         "Exton Elias"
32     ]
33 }
34 ]
35 }
```

Bonus - The other way around

- Creating a LabVIEW cluster typedef from a JSON file

```
1  {
2  "Actors": [{
3    "name": "Tom Cruise",
4    "age": 56,
5    "Born At": "Syracuse, NY",
6    "Birthdate": "July 3, 1962",
7    "photo": "https://jsonformatter.org/img/tom-cruise.jpg",
8    "wife": null,
9    "weight": 67.5,
10   "hasChildren": true,
11   "hasGreyHair": false,
12   "children": [
13     "Suri",
14     "Isabella Jane",
15     "Connor"
16   ]
17 },
18 ],
19 {
20   "name": "Robert Downey Jr.",
21   "age": 53,
22   "Born At": "New York City, NY",
23   "Birthdate": "April 4, 1965",
24   "photo": "https://jsonformatter.org/img/Robert-Downey-Jr.jpg",
25   "wife": "Susan Downey",
26   "weight": 77.1,
27   "hasChildren": true,
28   "hasGreyHair": false,
29   "children": [
30     "Indio Falconer",
31     "Avri Roel",
32     "Extton Elias"
33   ]
34 }
35 }
```



Cluster

Actors

0

name

age

Born At

Birthdate

photo

wife

weight

hasChildren

hasGreyHair

children

Bonus - The other way around

- Creating a LabVIEW cluster typedef from a JSON file

```
1  {
2  "Actors": [{
3    "name": "Tom Cruise",
4    "age": 56,
5    "Born At": "Syracuse, NY",
6    "Birthdate": "July 3, 1962",
7    "photo": "https://jsonformatter.org/img/tom-cruise.jpg",
8    "wife": null,
9    "weight": 67.5,
10   "hasChildren": true,
11   "hasGreyHair": false,
12   "children": [
13     "Suri",
14     "Isabella Jane",
15     "Connor"
16   ]
17 },
18 ],
19 "name": "Robert Downey Jr.",
20 "age": 53,
21 "Born At": "New York City, NY",
22 "Birthdate": "April 4, 1965",
23 "photo": "https://jsonformatter.org/img/Robert-Downey-Jr.jpg",
24 "wife": "Susan Downey",
25 "weight": 77.1,
26 "hasChildren": true,
27 "hasGreyHair": false,
28 "children": [
29   "Indio Falconer",
30   "Avri Roel",
31   "Exton Elias"
32 ]
33 }
34 }
35 }
```



Cluster

Actors

0

name

age

Born At

Birthdate

photo

wife

weight

hasChildren

hasGreyHair

children

OR

Cluster

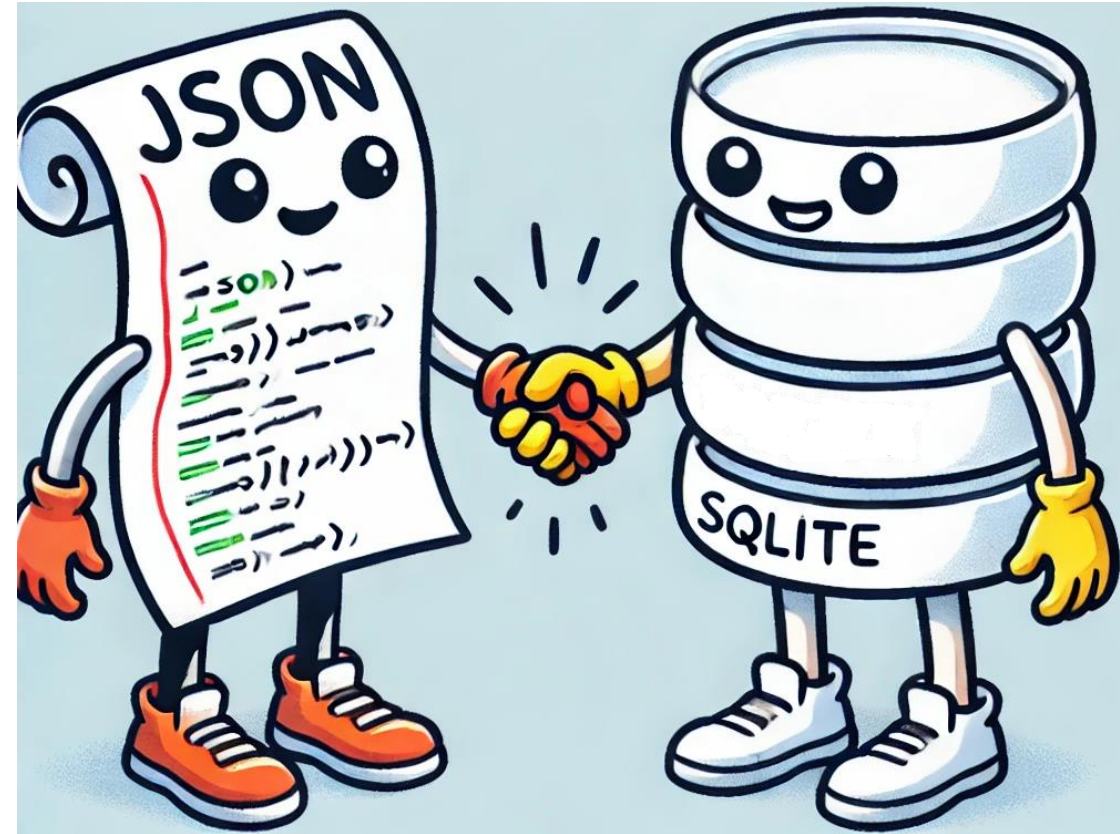
Actors

0

name	Tom Cruise	name	Robert Downey
age	56	age	53
Born At	Syracuse, NY	Born At	New York City, NY
Birthdate	July 3, 1962	Birthdate	April 4, 1965
photo	https://	photo	https://
wife		wife	"Susan Downey"
weight	67.5	weight	77.1
hasChildren	<input type="checkbox"/> OFF/ON	hasChildren	<input checked="" type="checkbox"/> OFF/ON
hasGreyHair	<input type="checkbox"/> OFF/ON	hasGreyHair	<input type="checkbox"/> OFF/ON
children	0	children	Indio Falconer

Takeaway

Convert your data into JSON and use SQLite to filter



Packages

- JSONtext by JDP Science
https://www.vipm.io/package/jdp_science_jsontext/
- SQLite Library by JDP Science
https://www.vipm.io/package/drjdpowell_lib_sqlite_labview/
- JSONtext Object Serialization by PNR
https://www.vipm.io/package/pnr_lib_jsontext_object_serialization/

Links

- Application Design Around SQLite - Dr James Powell - GDevCon#2
https://www.youtube.com/watch?v=i4_I-UuWtPY
- Tree Control – A Little Present
<https://forums.ni.com/t5/Random-Ramblings-on-LabVIEW/Tree-Control-A-Little-Present/ba-p/3486755>
- Encrypted SQLite – A Gift
<https://forums.ni.com/t5/Random-Ramblings-on-LabVIEW/Encrypted-SQLite-A-Gift/ba-p/4105019>

SQLite Tools

- JSON Validator <https://jsonlint.com/>
- JSON Path <https://jsonpath.com/>
- JSON Path Finder <https://jsonpathfinder.com/>

- DB Browser for SQLite <https://sqlitebrowser.org/>

- SQLite JSON Functions And Operators <https://www.sqlite.org/json1.html>

Special thanks to

- Dr James Powell
- Steve Watts
- Pascal Neuperger



A BIG
THANK
YOU!