CAMUNDA CON LIVE

JSON Handling in Camunda

Franz Flückiger, FROX AG

FROX
NOSERGROUP

JSON Handling in Camunda

CamundaCon 2021

23.09.2021 / Franz Flückiger

ABOUT ME

«Passionate about software development and music.»

Franz Flückiger

Software Consultant

ABOUT FROX

We are experts for digital solutions in service management and digital transformation, present on the Swiss market since 1996.

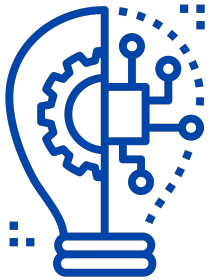
We develop tailor-made business process solutions with Camunda simplifying work for our customers.

FROX
NOSERGROUP

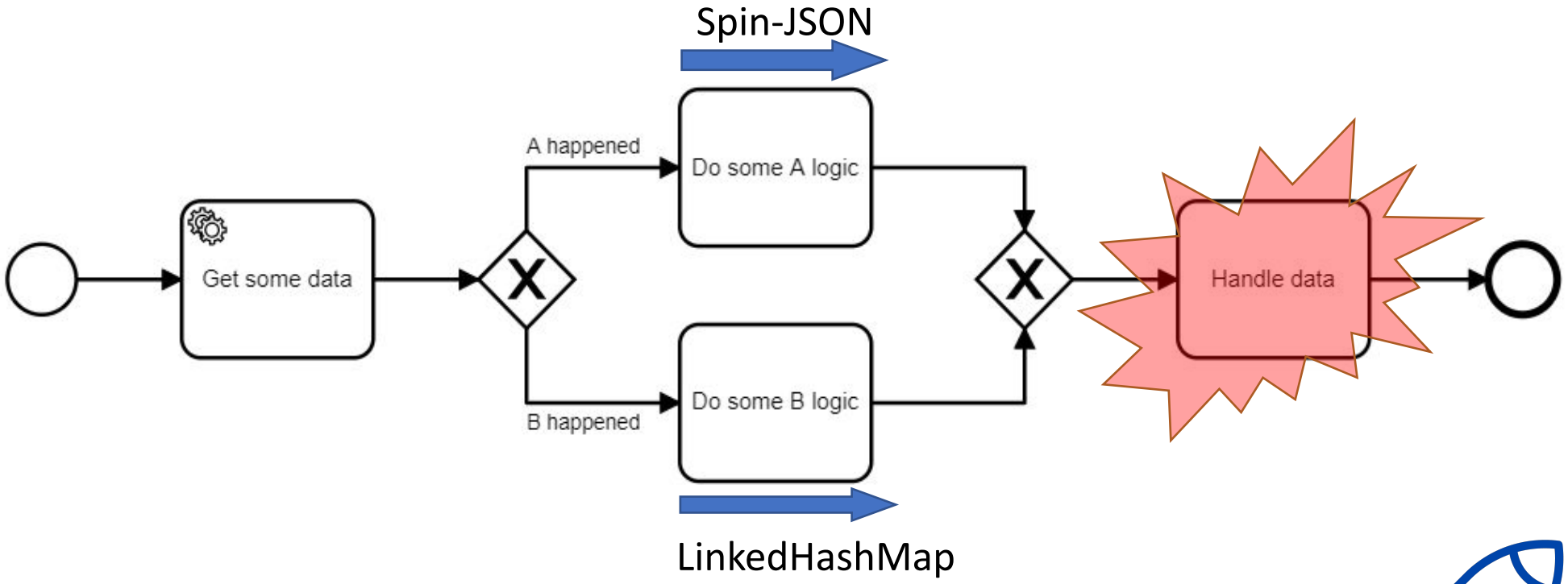
WHAT IS THIS PRESENTATION ABOUT?

👉 dpm-json

- New open-source tool to work with data in Camunda
- Speeding up process develpment
- Get it on https://github.com/frox-ag/dpm-json

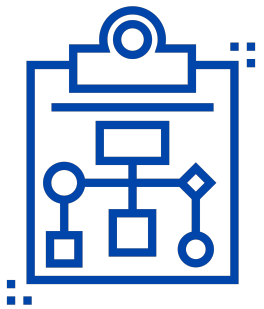


PREQUEL


Spin-JSON
A happened
Do some A logic
Get some data
Handle data
Do some B logic
B happened
LinkedHashMap

REQUIREMENTS

- Only one complex data type: Camunda’s Spin-JSON
- Intuitive JSON handling like Javascript
- Not breaking legacy processes
- Written in Groovy language



RESULT

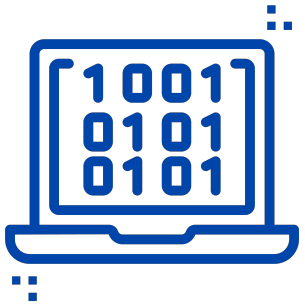
Name	Type	Value
myList	Json	[0,1,2,3]

Javascript with SPIN

```
var parsedList = JSON.parse(myList)

var filteredList = []
for(var it in parsedList) {
   if(it % 2 === 0) {
      filteredList.push(it)
   }
}
S(JSON.stringify(filteredList))

// returns [‘1’, ‘2’]
```

Groovy with dpm-json

```
dpmJson(myList).findAll { it.value() % 2 == 0 }

// returns [1, 2]
```

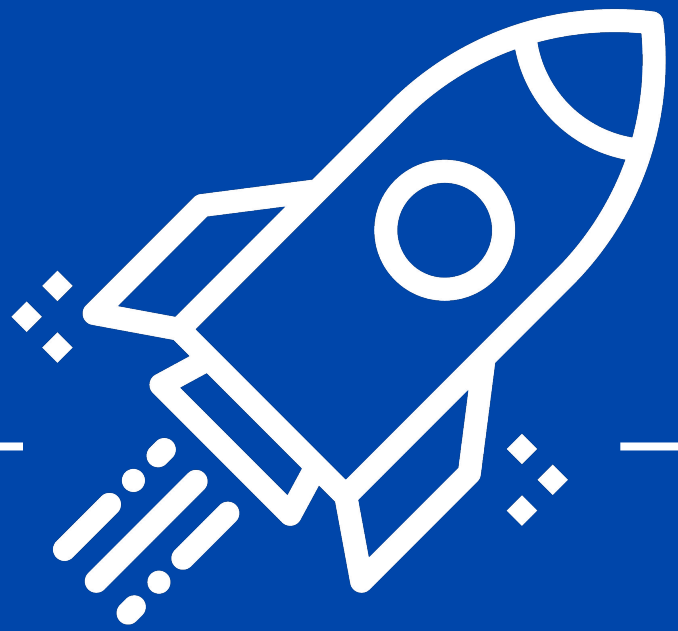


MISSION

Basic usage

List usage

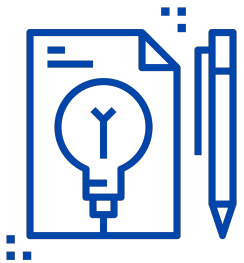
Map usage



GETTING STARTED

To access the dpm-json features simply type:

```
dpmJson(jsonObj)
```



INPUT DATA

- **ArrayList** (every class implementing java.util.List)

```
def myList = [0,1,2,3]
dpmJson(myList)
```

- **LinkedHashMap** (every class implementing java.util.Map)

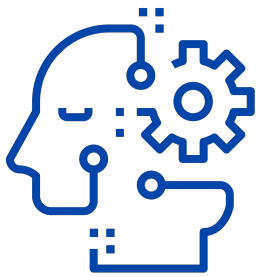
```
    def myMap = [myKey: "myValue"]
dpmJson(myMap)
```

- **JSON String**

```
    def myJsonString = '{ "myKey": "myValue" }'
dpmJson(myJsonString)
```

- **Spin Json Objects**

```
dpmJson(mySpinJsonObject)
```



OUTPUT DATA

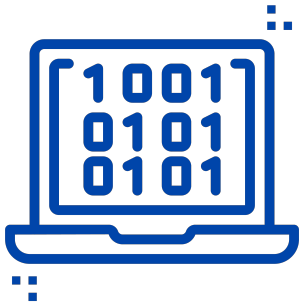
Always SPIN-JSON!

ATTRIBUTE ACCESS

```
def myObj = dpmJson(jsonObj)
myObj.myKey
```

or

```
myObj['myKey']
```

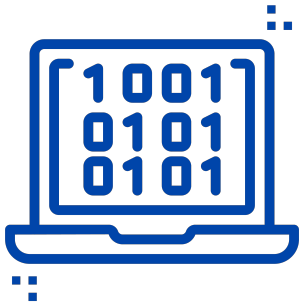


VALUE ACCESS

```
def myObj = dpmJson(jsonObj)
myObj.myKey.value()
```

or

```
myObj['myKey'].value()
```

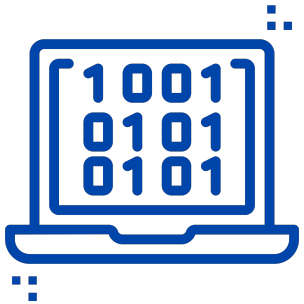


VALUE ASSIGNMENT

```
def myObj = dpmJson(jsonObj)
myObj.myKey = ’myNewValue’
```

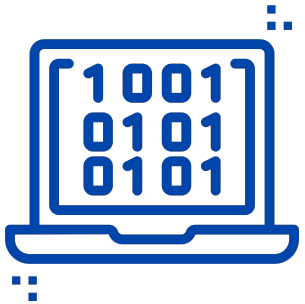
or

```
myObj[’myKey’] = ’myNewValue’
```



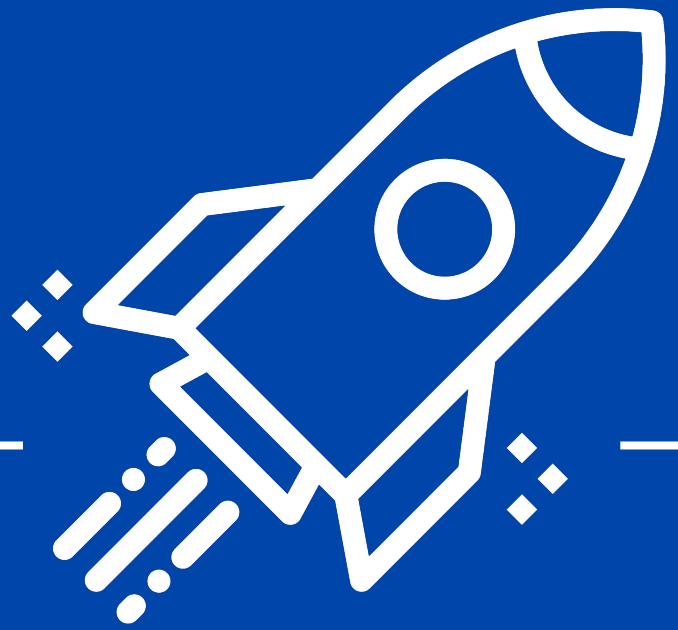
VERIFYING ATTRIBUTE EXISTENCE

```
def myObj = dpmJson()

myObj.myKey.exists()
// returns false
```



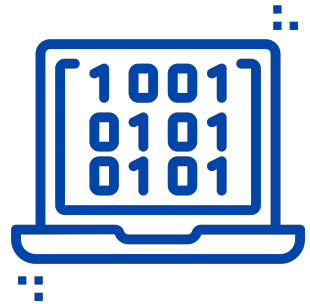
MISSION

- Basic usage
- List usage
- Map usage



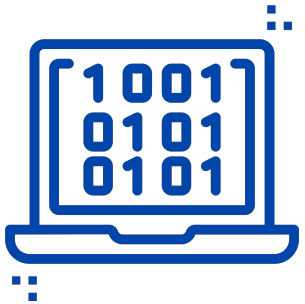
LIST OPERATIONS - BASICS

```
def myList = dpmJson([0,1,2,3])
myList.isList()

myList.size() // returns 4

myList.push() // adds a new object to the list

myList[2] // returns the third element
```



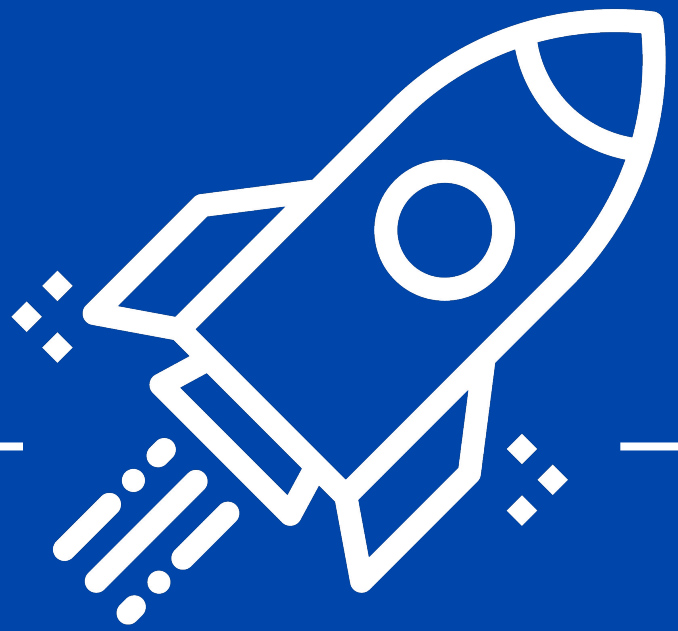
LIST OPERATIONS - LOOPS

```
myList.each { println(it) }

myList.sort { it.myKey.value() }

myList.findAll {it.value() % 2 == 0 }
```



MISSION

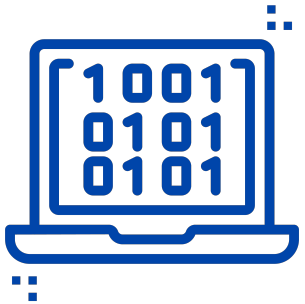
- Basic usage
- List usage
- Map usage



MAP OPERATIONS – CHECK IF MAP

Usage similar to lists!

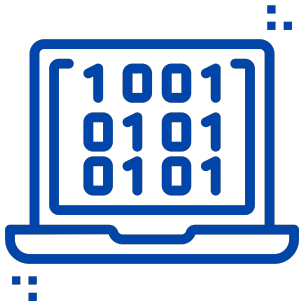
```
def myMap = dpmJson([a: 1, b: 2])

myMap.isMap()
// returns true
```



MAP OPERATIONS – LOOPS

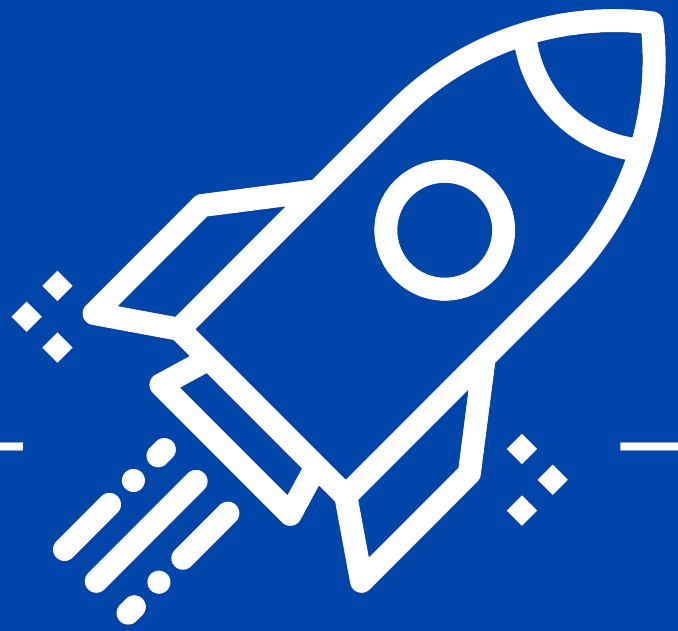
```
def myMap = dpmJson([a: 1, b: 2])

myMap.each { it }
// it is composed by it.key which is a String
// and it.value which is the wrapped value
```

it.value ≠ it.value()



MISSION

- Basic usage
- List usage
- Map usage



LIVE DEMO

Let's see this live!

GET IT!

https://github.com/frox-ag/dpm-json

QUESTIONS?

CONTACT

franz.flueckiger@frox.ch

Franz Flückiger
Software Consultant
FROX


FROX
NOSERGROUP
www.frox.ch

LIST OPERATIONS - LOOPS

```
def myList = dpmJson([0,1,2,3])

for(it in myList) {
    ... your logic here ...
}
```