redislabs
HOME OF REDIS

Possibilities for Rules Engine Replacement

May 2019 | Andre Srinivasan

Recap

- Current rules engine is pretty big
  - Java
  - Drools

- Requirements (as I understand them)
  - Small runtime footprint
  - IFTTT
  - Stateless declarative rules
  - Durable

Options Found

- Durable Rules Engine
  - Engine written in C
  - Python / Node.js / Ruby / JSON

  "...Micro-framework for real-time, consistent and scalable coordination of events...

  ...Full forward chaining implementation (A.K.A. Rete) to evaluate facts and events in real time...

  ...Define simple and complex rulesets as well as control flow structures such as flowcharts, statecharts, nested statecharts and time driven flows..."

- JsonLogic
  - Go implementation(!)

  "...It's a small, safe way to delegate one decision…

  ...The rule is data, you can even build it dynamically…

  ...No setters, no loops, no functions or gotos. One rule leads to one decision, with no side effects and deterministic computation time…."

Durable Rules Engine

```
with ruleset('motortoofastsignal'):
   @when_all((m.device == '562114e9e4b0385849b96cd8') & (m.parameter == 'RPM') & (m.value > 1200))
   def say_hello(c):
       print ('Hello {0} warning triggered for high engine speed'.format(c.m.device))


run_all()
```

JsonLogic

```
  _ = json.Unmarshal([]byte(`{
    "filter": [
      { "var": "events" },
      { "and":
     { "=": [{ "var": "device" },"562114e9e4b0385849b96cd8"],
          ">": [{ "var": "RPM" }, 1200]
       }}]}`), &logic)

  _ = json.Unmarshal([]byte(`{
    "events": [{"device": "562114e9e4b0385849b96cd8", "RPM": 1200}, {"device": "562114e9e4b0385849b96cd8",
"RPM": 1201}, {"device": "111", "RPM": 1200}, {"device": "222", "RPM": 1201}
    ]}`), &data)

  err := jsonlogic.Apply(logic, data, &result)
```

redislabs
HOME OF REDIS