Decoupling your Spring boot Microservices

With an open source workflow engine
Orchestrating Spring Boot Microservices
(In order to get coffee quickly)

With an open source workflow engine
A Classic Distributed System

Service One

Rest

Response

Service Two
A Classic Distributed System
Congratulations! You’ve got a coffee

Who’s next?
Things to think about?

- Barista takes a break
- Order is taking while
- “We don’t do mochachinos”
Starbucks does not use two-phase commit

Gregor Hohpe
https://www.enterpriseintegrationpatterns.com/ramblings/18_starbucks.html
The System of Kaffee und Kuchen

Cashier: Takes orders

Order

Order Sorter: Can get Cake!

Coffee

Barista: Makes Coffee

Complete

Coffee Ready
Demo!

https://github.com/NPDeehan/kaffee-und-kuchen
Let's Make some changes to our system
What needs to change?

We Need to Hold State

- Retry
- Timeout
- Callback to Cashier
- Versioning
- Error Handling
Live coding!

https://github.com/NPDeehan/kaffee-und-kuchen
Let's deal with the issues

- What happens if the barista is taking a quick break?
  - Hold State until the barista is back
- What happens if the order is taking too long?
  - Let the customer know we're still working on it
- What happens if the order can't be completed for business reasons - not technical reasons?
  - BPMN Error event!
Things to think about?

- Barista takes a break
- Order is taking while
- “We don’t do mochachinos”
Live coding!

https://github.com/NPDeehan/kaffee-und-kuchen
What did we Achieve

• Better throughput
• Better error handling
• Scale better
• Visualization
• Timer(?)
• Saga(?)
What needs to change?

We Need to Hold State

- Retry
- Timeout
- Callback to Cashier
- Versioning
- Error Handling
Visualization!
start.camunda.com
Typical conclusion of a technical talk

Speaker: Questions?
...
Embarrassing silence
...
Speaker: ok, I'm done

Applause

Attendees approach speaker to ask questions privately

Please don't be shy. If answer is interesting for you, very likely it will be for many others

#Devoxx