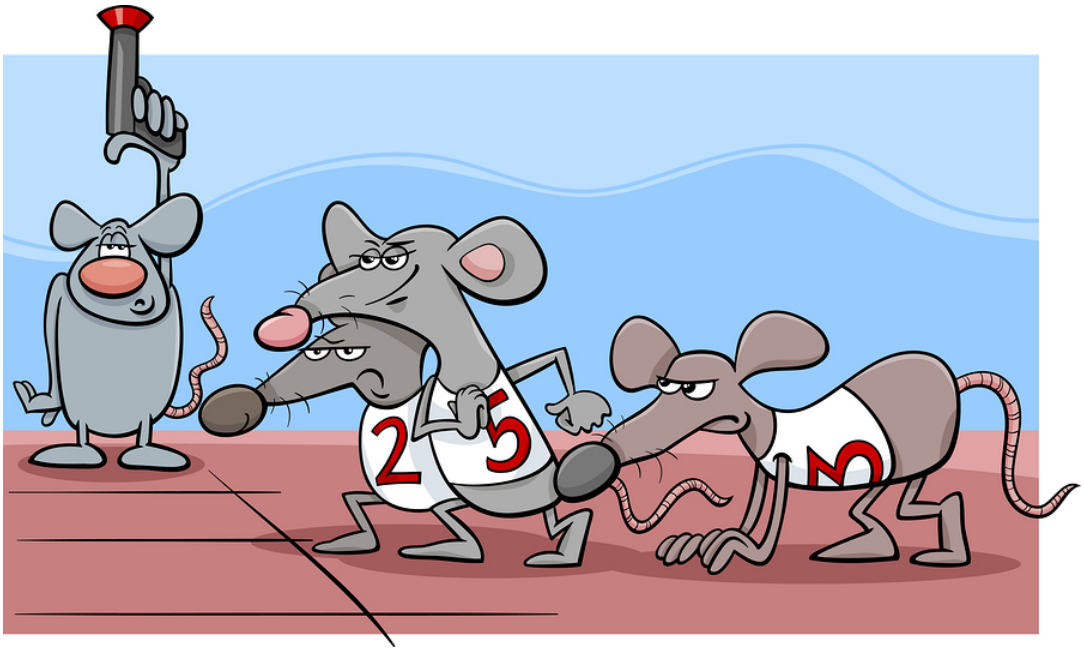
CC :: SOMKIAT

Part 1 :: ความรู้พื้นฐานเกี่ยวกับ Reactive Programming

somkiat | October 8, 2019 | Programming | No comments



จากทางกลุ่ม RxJS Thailand ทำการ share บทความ
เกี่ยวกับ **The introduction to Reactive Programming you've been missing**
อ่านแล้วน่าสนใจดี สำหรับคนเริ่มต้นใหม่ ๆ แบบผม
เลยทำการสรุปไว้นิดหน่อย

Reactive Programming คืออะไร ?

ในบทความอธิบายไว้ว่า ความหมายของ Reactive Programming ในแต่ละที่
มักจะอธิบายไม่ค่อยเข้าใจเท่าไร
ทั้งที่ Wikipedia จะออกแนวทฤษฎีมากไป
ทั้งที่ Stackoverflow นี่ยิ่งไม่เหมาะกับผู้เริ่มต้นใหม่
ทั้งที่ Reactive Manifesto นี่พูดภาษา business และ management มาก ๆ
ทั้งที่ทาง Microsoft นี่ก็ผูกติดกับเครื่องมือและ framework ของตนเองเกินไป

> *ดังนั้นในบทความจึงทำการอธิบายไว้ว่า*
> ***Reactive programming is programming with asynchronous data streams.***

Searc

E-Book
- [Tips] การใช้งาน Postman สำหรับทดสอบ APIs
- 20 สิ่งที่นักพัฒนา software ควรรู้
- สรุปจาก course Practical Go 2019
- หนังสือมือสอง

Categories
- Architecture
- BigData
- Book
- course
- data-science
- dev-ops
- Meetup
- Practice
- Programming
- Review
- Tools
- เรื่องทั่วไป

Tags
97-thing-developer-should-know agile android android-studio architecture automated-test bad-code

นั่นคือการ programming กับ stream ของข้อมูลที่เป็นแบบ asynchronous
ดังนั้นเราสามารถ observe
หรือคอยเฝ้าสังเกตหรือจับตามองข้อมูลหรือ event เหล่านั้น
จากนั้นจึงทำงานตามข้อมูลเหล่านั้น (**side effect**)
การทำงานนั้น ๆ ก็ก่อให้เกิดเหตุการณ์อื่น ๆ ขึ้นมา
วนไปเรื่อย ๆ

> *เป็นแนวคิดเดิม ๆ ไม่มีอะไรใหม่เลย*
> *เนื่องจากทุกครั้งที่เกิดเหตุการณ์ใด ๆ (Event)*
> *มักจะก่อให้เกิดการเปลี่ยนแปลงเสมอ ไม่มากก็น้อย (Side effect)*

โดยที่ใน data stream เป็นข้อมูลอะไรก็ได้

ยกตัวอย่างเช่น

- User input
- Variable
- Property
- Cache
- Data structure ต่าง ๆ

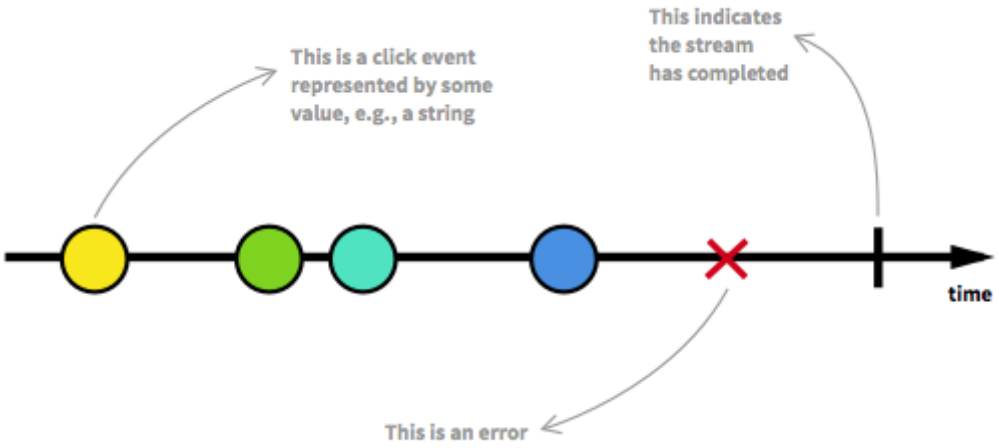
Stream นั้นมี function ทำการงานพื้นฐานเช่น

- การสร้าง stream
- การกรองหรือ filter ข้อมูลใน stream
- การรวมหรือ combine ข้อมูลใน stream

Stream ก็เหมือนท่อน้ำที่เราสามารถตัดต่อ รวม หรือกรองได้เลย
เพื่อให้ทำงานตามที่เราต้องการ

โดยที่ Stream คือส่วนการทำงานหลักของ Reactive เลย

แสดงดังรูป



จากรูปนั้น เป็นข้อมูลของเหตุการณ์ต่าง ๆ ที่เกิดขึ้น
ถูกเรียงตามเวลาที่เกิดขึ้นของแต่ละเหตุการณ์
โดยเหตุการณ์จะมีอยู่ 3 กลุ่มคือ

1. **Value หรือ Type** เรียกง่าย ๆ คือ เหตุการณ์อะไรนั่นเอง เช่น Clicked, Created user เป็นต้น
2. **Error** คือเมื่อเกิด error หรือข้อผิดพลาดจากเหตุการณ์ต่าง ๆ
3. **Completed** คือเมื่อทำงานสำเสร็จจากเหตุการณ์ต่าง ๆ

เหตุการณ์สามารถเกิดขึ้นได้ตลอดเวลา

ส่วน **Error** หรือ **Completed** ของแต่ละเหตุการณ์
จะเกิดตอนไหนก็ได้หลังจากเหตุการณ์นั้น ๆ เกิดขึ้น
นั่นคือ เราไม่สามารถเดาได้เลยว่าจะเกิดขึ้นตอนไหน (ทำงานแบบ Asynchronous)

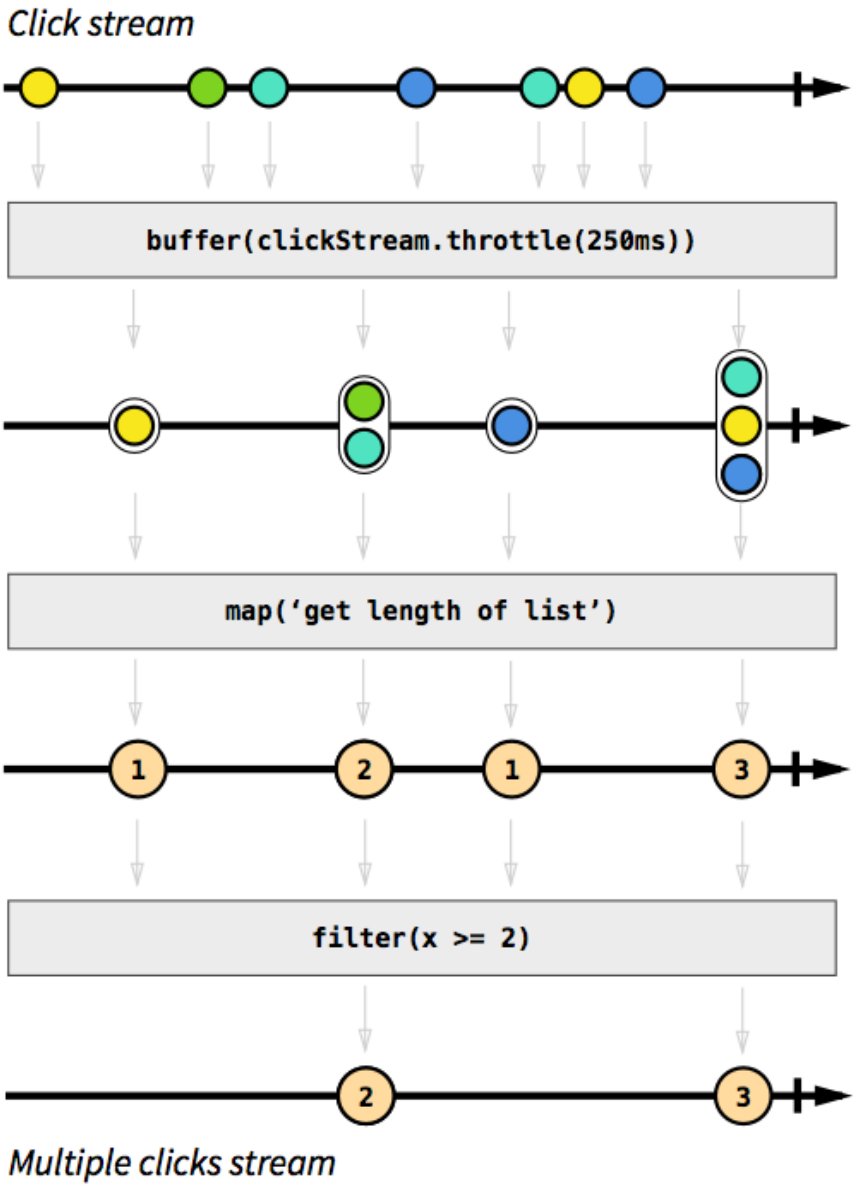
ดังนั้นเราจะเฝ้ามองหรือดักฟังเหตุการณ์ต่าง ๆ เหล่านี้ได้อย่างไร ?
ส่วนใหญ่เรามักจะได้ยินว่า
เราจะดักฟังเหตุการณ์ที่เกิดขึ้นใน stream ได้จะต้องทำการ **subscribe** ก่อนเสมอ
มันคือการทำงานตาม **Observer Design Pattern**

ในบทความนั้นทำการยกตัวอย่างที่น่าสนใจคือ

เพื่อทำให้เข้าใจ function การทำงานพื้นฐานของ Reactive library
เช่น map, filter และ scan เป็นต้น
ทำการเก็บข้อมูลของการ click ใน data stream
จากนั้นจะทำการตรวจหา **multiple-click** กัน โดยแนวทางการคิดเป็นดังนี้

- ทำการตรวจสอบในแต่ละเหตุการณ์ว่าเกิดขึ้นใกล้เคียงในช่วงเวลาที่กำหนดหรือไม่ เพื่อรวมกลุ่มกัน
- ทำการนับจำนวนในแต่ละกลุ่ม
- ทำการกรองเฉพาะกลุ่มที่มีจำนวนมากกว่า 1 ขึ้นไปเท่านั้น

แสดงการใช้งาน function จาก Reactive Library ดังรูป
จะเห็นได้ว่าเป็นแนวคิดที่เรียบง่ายมาก ๆ



ใน Part ต่อไปเรามาคิดเชิง Reactive Programming
และตัวอย่าง code กัน
ขอให้สนุกกับการ coding ครับ

Tags: reactive-programming

Article by Somkiat Puisungnoen

To be Craftmanship

Related Posts

รูปอธิบายเรื่อง Eventual Consistency

ว่าด้วยเรื่องของ Functional Programming :: Thinking

differently about problems

ว่าด้วยเรื่อง Testing in Production (TiP)

แปลและสรุปเรื่อง Practical Persistence in Go: Organising Database Access

สรุปเรื่อง The Seven Wastes ในการพัฒนา Software

18 ปีของ Spring Framework โดยคุณ Rod Johnson