

# FastRxJava

이승민

뱅크샐러드 Android Developer / GDE Android Korea

[rfrost77@gmail.com](mailto:rfrost77@gmail.com)



# 쓰레드 관리

# RxJava 사용처

State 관리

다중 비동기 처리

이벤트 처리

# RxJava 사용처

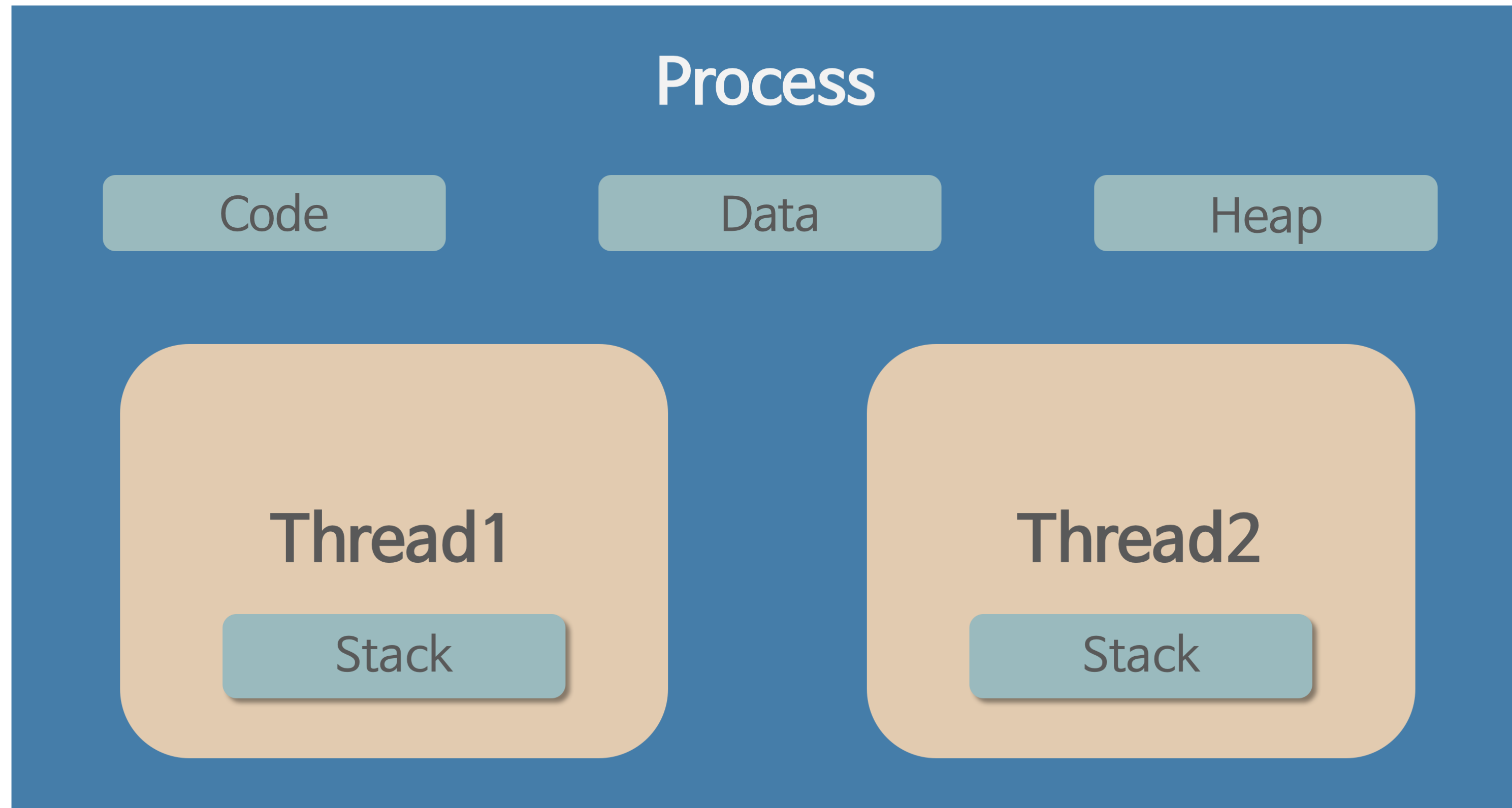
State 관리

**다중 비동기 처리**

이벤트 처리

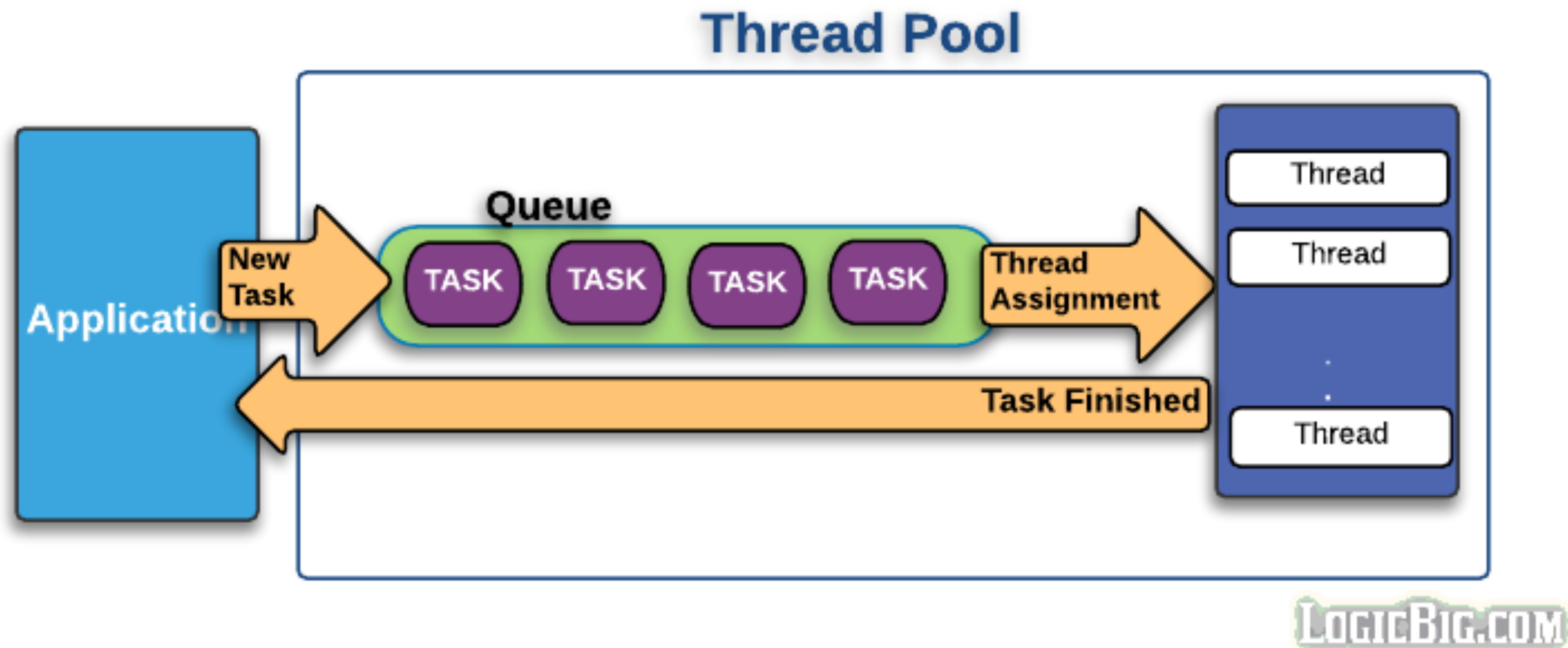
**프레드**

# 프로세스 vs 스레드



# 프로그램 vs Worker

# 스레드풀



스레드 **재사용** 바꾸니

# Java 스레드풀 ExecutorService

cached  
fixed  
single

# RxJava 스레드 or 풀

main

io

computation

trampolin

newThread

single

from



# Main 스레드

UI 그리기  
다른 작업 불가

비동기 처리 시 다른 스레드 활용

**io 프레임**

**가장 많이 사용  
io 작업할 때 사용**

**computation 프레임드**

**계산 등 CPU 많이 사용하는 프레임드**

**trampolin 프레임**

**현재 프레임 그대로 이어감  
주로 다형성으로 테스트에서 사용**

# RxJava 스레드 관리

**subscribeOn**

**observeOn**

# RxJava 스레드 관리 - zip/merge

여러 동작을 병행 수행

# RxJava 스레드 관리 - zip/merge

여러 동작을 병행 수행

**subscribeOn!!**

# Stetho

## Let's 디버깅



# 문제풀이


<https://lorentzos.com/i-bet-your-rxjava-is-on-the-wrong-thread-ae02e66a3eac>






구독




**subscribe()**

# 1. Observable chain 생성 - source, downstream 넘기기

1. **Observable chain 생성 - source, downstream 넘기기** 
2. **Observable.subscribe()**

1. **Observable chain 생성 - source, downstream 넘기기** 
2. **Observable.subscribe()**
3. **Observable.subscribeActual(t: downstream)**

1. **Observable chain 생성 - source, downstream 넘기기** 
2. **Observable.subscribe()**
3. **Observable.subscribeActual(t: downstream)**
4. **source.subscribe() - 넘어온 source 따라 올라가 실행** 

1. **Observable chain 생성 - source, downstream 넘기기** 
2. **Observable.subscribe()**
3. **Observable.subscribeActual(t: downstream)**
4. **source.subscribe() - 넘어온 source 따라 올라가 실행** 
5. **downstream.onSubscribe() - observer를 upstream set** 



1. **Observable chain 생성 - source, downstream 넘기기** ⬇️
2. **Observable.subscribe()**
3. **Observable.subscribeActual(t: downstream)**
4. **source.subscribe() - 넘어온 source 따라 올라가 실행** ⬆️
5. **downstream.onSubscribe() - observer들 upstream set** ⬇️
6. **downstream.onNext(downstream) - downstream 따라 내려가 실행** ⬇️




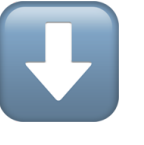
1. **Observable chain 생성 - source, downstream 넘기기** ⬇️
2. **Observable.subscribe()**
3. **Observable.subscribeActual(t: downstream)**
4. **source.subscribe() - 넘어온 source 따라 올라가 실행** ⬆️
5. **downstream.onSubscribe() - observer를 upstream set** ⬇️
6. **downstream.onNext(downstream)** - downstream 따라 내려가 실행 ⬇️

\* upstream = Source - 이전 Observable들

\* downstream - 이후 Observer들

\* isDisposed() - (upstream == disposed)

\* upstream.subscribe() -> downstream.onNext()

1. **Observable chain 생성 - source, downstream 넘기기** 
2. **Observable.subscribe()**
3. **Observable.subscribeActual(t: downstream)**
4. **source.subscribe() - 넘어온 source 따라 올라가 실행** 
5. **downstream.onSubscribe() - observer들 upstream set** 
6. **downstream.onNext(downstream) - downstream 따라 내려가 실행** 

\* upstream = Source - 이전 Observable들

\* downstream - 이후 Observer들

\* isDisposed() - (upstream == disposed)

\* upstream.subscribe() -> downstream.onNext()

# Operators

Observable Chain

Observable - Observer 1:1 Mapping

Decorator Pattern

downStream에 downStream을 씌우기

끝

이승민

뱅크샐러드 Android Developer / GDE Android Korea

[rfrost77@gmail.com](mailto:rfrost77@gmail.com)