Functional Programming with Scala: Class 7

John Nestor
47 Degrees, Persist Software

uw@persist.com

November 28, 2016

Review

- Trains
- Present solution(s)
- Code review and discussion

Lecture

ScalaCheck

- Nice intro talk by Noel Markham of 47 Degrees
- http://noelmarkham.github.io/practical-scalacheck/index.html#/

Futures

- Chapter 2
- Future.sc
- have map flatMap filter(not useful)

Execution Contexts

- From
  - Import default
  - more options in Akka
- What: scheduler
  - ec < Thread < Process
  - multiplexed Thread pool (futures, actors, scheduler)
- How
  - Keep body short
  - Don’t block (ties up thread), Don’t use Await,
  - Don’t share mutable state across futures
- Missing: Concurrency control
  - Could use Java synchronize, …
  - Better to use Akka actors

Async Futures

- f(a:B):Future[B]
  - Non blocking delays (Akka scheduler)
  - Database queries (JDBC SQL problem)
  - I/O
  - Http client
  - Http server

Future Exceptions

- Future are like Try. Both can carry an exception.
- FutureExcept.sc

Future: Fork Join

- ForkJoin.sc

Promises

- Promises operations are thread safe
- Promise.sc

Assignment 7

Assign 6: Trains

- Goals
  - Another pure functional exercise
  - Graph data structures
  - Graph algorithm