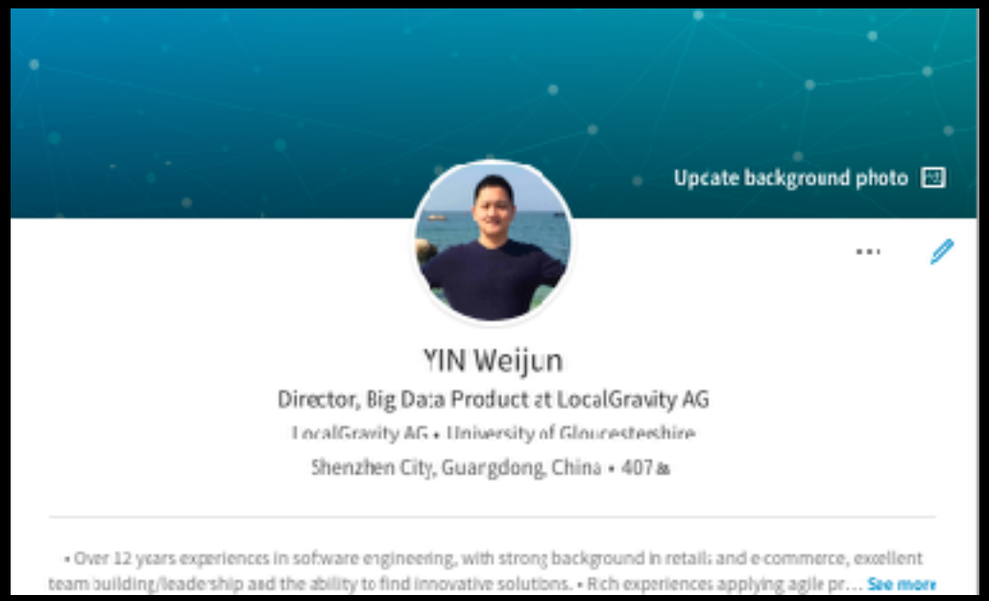
Actor Model

What it means to me



尹伟君

linkedin.com/in/yin-weijun-93a08816/

Agenda

- Actor Model - actor, communication, address
- Confusing terms - Server, Supervisor, Application

Duck model

Persistent - 指的是一旦被创建它被保存下来，直到收到shutdown message才会销毁；对比thread还有future这点不一样。

Encapsulate - 唯一可以了解internal state的方法，是通过send actor a message。

什么是Actor Model?

A method of concurrency in which the universal primitive is an actor

A mathematical model to understanding concurrent computation.

To me, it’s a conceptual tool for understanding how it works and comparing to other concurrent models.

It was inspired by physics;
Everything is an actor; in Elixir/erlang, process is actor.

Erlang does not appear to have been directly influenced by work on the Actor model.
Erlang is NOT an implementation of the Actor model.

Actor

角色

什么是Actor?

Actor是最小的组成单元，在Actor model中，任何事物都可以使用actor来代表

Everything is actors

“The primitive unit that embodies three essential elements of computation - processing, storage, communication”

–Hewitt

Processes
Threads (system or green)
Futures
Coroutines
CSP
Petri nets
etc.

Actor 特征

- Actors are persistent
- Encapsulate internal state
- Actors are asynchronous

Cited from [1]

Embody three things: Processing, storage, communication

Persistent - 指的是一旦被创建它被保存下来，直到收到shutdown message才会销毁；对比thread还有future这点不一样。
Encapsulate - 唯一可以了解internal state的方法，是通过send actor a message。
Asynchronous - Actor间的沟通是异步的

What Actor Can Do

- Create more other actors
- Receive messages and in response:
  - Make local decisions
  - Perform arbitrary, side-effecting action
  - Send messages to the address it knows
  - Respond to the sender 0 or more times
- Process exactly one message at a time

Cited from [1]

What Actor can do - 公理，或者可以理解为游戏规则。

⚠️：actor是对message被动响应的:

给自己发message，然后改变内在状态；

Side effect例子：日志

Concurrency does not occur inside process. There is a queue called mailbox inside each process.

Persistent - 指的是一旦被创建它被保存下来，直到收到shutdown message才会销毁；对比thread还有future这点不一样。

Encapsulate - 唯一可以了解internal state的方法，是通过send actor a message。

Communication 特征

- No channel or intermediaries
- “Best effort” delivery
- At-most-once delivery
- Messages can take arbitrary long to be delivered
- No message ordering guarantees

Cited from [1]

Messages are all equal priority.

例子：漂流瓶可能会100年后经辗转，最终送到...

No ordering guarantees, but you can implement your own ordering. e.g. an actor/process acts as channel receiving message with order number.

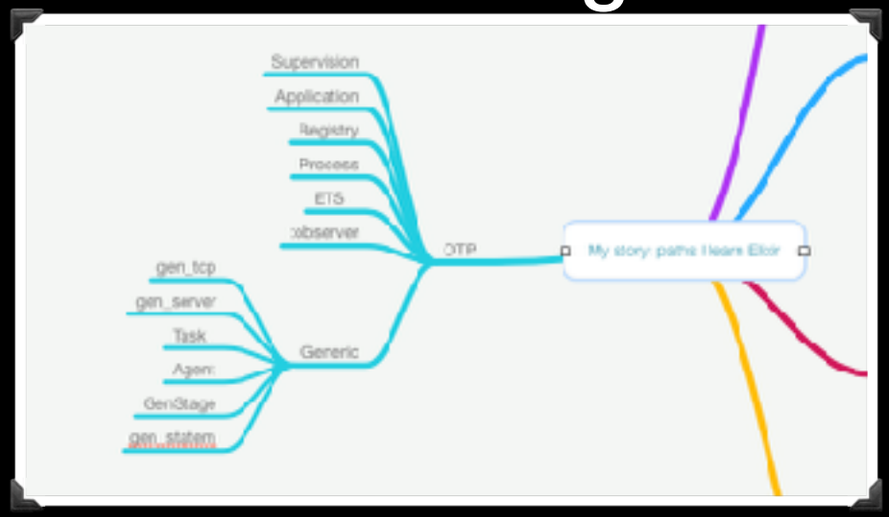
Address

- Identifies an Actor
- May also represent a proxy / forwarder to an Actor
- Contains location and transport information
- Location transparency

Cited from [1]

Address is right word for capability this point.

More Confusing Terms



- Server (别名: 框架), is a kind of process
- Supervisor (别名: 保姆), is a kind of process
- Component is a better name for Erlang application

1. Elixir server. Like Template design pattern; we can build on top of it fits into our own purpose.
   Why not separate the actual imp from the server code/abstraction??

2. Erlang approach to deal with fault-tolerance concern；
   Process Tree vs Supervision Tree

3. To enable auto start up, like main() in java;

Application 更多指的是在客户端运行的程序；erlang application更像可重用性的component，由一组processes一起解决某个关注的问题。
Application, has own state; code runs in own process.
Library uses client’s state; code runs in client process.

References

- [1]: https://www.youtube.com/watch?v=lPTqcecwkJg
- [2]: https://www.youtube.com/watch?v=7erJ1DV_Tlo
- [3]: https://pragprog.com/book/pb7con/seven-concurrency-models-in-seven-weeks
- [4]: http://erlang.org/pipermail/erlang-questions/2014-June/079860.html