Akka学习笔记：Actors介绍

Akka学习笔记系列文章：
《Akka学习笔记：ACTORS介绍》
《Akka学习笔记：Actor消息传递(1)》
《Akka学习笔记：Actor消息传递(2)》
《Akka学习笔记：日志》
《Akka学习笔记：测试Actors》
《Akka学习笔记：Actor消息处理-请求和响应(1) 》
《Akka学习笔记：Actor消息处理-请求和响应(2) 》
《Akka学习笔记：ActorSystem(配置)》
《Akka学习笔记：ActorSystem(调度)》

任何在过去做过多线程开发的人都不会否认维护多线程应用程序是多么难和头疼的一件事！我这里说的是维护，因为多线程开发开始的时候很简单，当你看到性能的提升对你来说是件多么高兴的一件事。然而，当你看到在子任务中很难找到容易的方法来从错误中恢复；或者是存在僵尸进程的bug很难重现；或者你的监控程序显示你的线程浪费大量的时间来等待共享状态而阻塞的时候对你来说是多么头疼！

我在这里并没有提到Java的并发API和它的集合使得多线程的编程变得相当简单轻松，因为我相信你既然来到这里，这就说明你希望能更好地控制你的子任务，或者你就是不喜欢使用锁以及同步块，希望能有一种更高层次的抽象。

在本系列的Akka笔记中，我将会通过简单的Akka例子来探讨Akka的多种特性。

什么是ACTORS

Akka Actors遵循Actor模型(废话！)

我们这把Actors当作是一个人，这个人不会自己和其他的人直接说话，他们只通过mail来进行交流。

现在来探讨Actors的一些特性：

一、消息传递

假设有两个人：学生和聪明的老师。学生每天早上都会给老师发送邮件，而聪明的老师都会回复一句名言。这里需要解释：

1、学生发送邮件。一旦发送成功，邮件不能再修改。这天然就具备了不可变性；
2、老师会自己决定何时检查邮箱；
3、老师还会回复一封邮件（也是不可变的）；
4、学生会自己决定何时检查邮箱；
5、学生不会一直等待回信（非阻塞的）

这就可以总结出Actor模型的一个基本特征——消息传递

二、并发

现在，假设有三个聪明的老师和三个学生。每个学生都会给每个老师发送邮件。这会发生什么事？其实什么都
没改变！每个人都有他自己的邮箱。
这里需要注意的一点：
默认情况下，邮箱里面的邮件是按照他们先后达到的次序进行阅读和处理的。

本质上，这很像是ConcurrentLinkedQueue。没有人去等待邮件被阅读，简单来说这就是一个非阻塞的消息（在Akka中内置了许多的mailboxes，这里http://doc.akka.io/docs/akka/snapshot/scala/mailboxes.html，包括了有界和基于优先级的。其实，我们自己也可以去实现）。

三、错误恢复

假如这三个老师分别来自不同的院系：历史系、地理系和哲学系。

历史系的老师用过去的某个事件笔记进行回复；而地理系的老师回复了一个有趣的地点；哲学系的老师回复了一个引用。每个学生分别给每个老师发送消息并分别得到回复。学生并不关心邮件到底是系里的哪个老师回复的。如果有一天有个老师生病了呢？系里至少得有一个老师在处理邮件才行。这样的话，系里的另一位老师就会顶上这项工作。

这里需要注意的地方：

1、会有一个Actor池，每个Actor会处理不同的事件。

2、Actor做的事情可能会抛出异常，而它自己无法从中恢复。在这种情况下，需要再生成（created）一个新的Actor来顶替它。换句话说，这个新的Actor会忽略刚才那条消息，继续处理剩余的消息。这些也被称为指令（Directive），后面我们会再讲到它们。

四、多任务

假设学生需要考试成绩，每个老师是通过邮件来发送的。也就是说，Actor可以处理多种类型的消息。

五、消息链

假如学生只想收到一封邮件而不是三件呢？

我们也可用用Actor来实现！我们可以通过分层来把老师连在一起。这个我们将在后面讲到Supervisor和Future的时候再回来讲。

应Mohan的要求，我们把类比的实体和Actor模型中的组件做一下映射。

学生和老师变成我们的Actor，Email Inbox对应Mailbox 组件。请求和响应不可修改、它们是不可变对象。最后，Actor中的MessageDispatcher组件将管理mailbox，并且将消息路由到对应的Mailbox中。

待续。。

本文原文地址：http://rerun.me/2014/09/11/introducing-actors-akka-notes-part-1/