عنوان: **معرفی Actor Based Programming و توسعه نرم افزار های مقیاس پذیر و دارای عملیات همزمان بسیار زیاد - قسمت اول**
نویسنده: ایمان رحیمی نیا
تاریخ: ۱۳۹۴/۰۵/۱۴ ۹:۲۰
آدرس: www.dotnettips.info
گروه‌ها: Asynchronous Programming, Tpl DataFlow, Erlang, Akka

مقدمه :

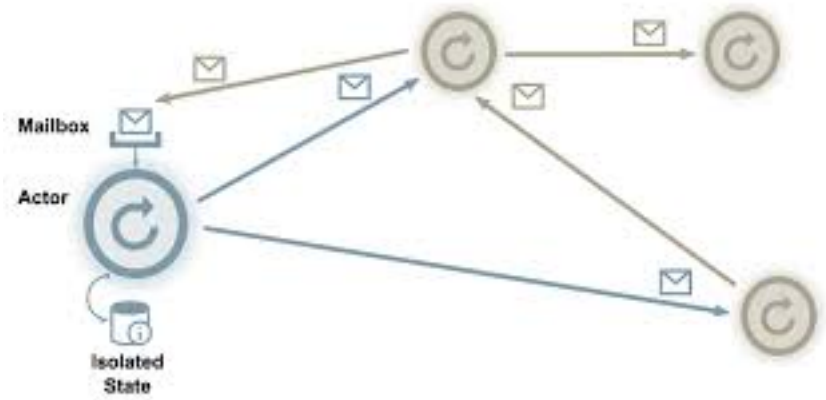
زمانیکه هدفمان تولید سامانه‌ی نرم افزاری باشد که تعداد بسیار زیادی از کاربران با آن سرو کار دارند و اتفاقاً این سامانه قرار است عملیات بسیار حساسی (نظیر عملیات بانکی و مالی، مخابراتی و ...) را انجام دهد و عدم سرویس دهی مناسب آن قابل تحمل نبوده و باعث خسارات مالی، نارضایتی و ... گردد می‌بایست از روش‌های خاصی برای توسعه‌ی این گونه سیستم‌ها استفاده نمود. این نرم افزارها برای اینکه بتوانند به تعداد درخواست‌های بسیار زیاد همزمان پاسخگو باشند و سرویس خود را با کیفیت مناسب ارائه دهند، می‌بایست دارای ویژگی‌های خاصی نظیر مقیاس پذیری (scalable) و تحمل پذیری در مقابل خطا (fault tolerance) باشند.

خوب حالا که صورت مسئله مشخص شد، یکی از راه حل‌های موجود را که مدل Actor Based است، بررسی می‌کنیم.

مدل Actor Based یکی از مدل های استاندارد برای توسعه‌ی نرم افزارهایی با قابلیت اطمینان بسیار بالا، تحمل پذیر درمقابل خطا و پاسخ دهی بسیار سریع می باشد. در این مدل، وظایف نرم افزار به مجموعه‌ای از Actor ها تقسیم (توزیع) گردیده و هر یک از Actor ها به صورتی کاملاً ایزوله، در نخ(thread) خاص خودشان اجرا شده و بخشی از وظایف سیستم را انجام می دهند. سپس با اتصال Actor ها به یکدیگر، یک خط لوله (Pipeline) تشکیل شده و با استفاده از مکانیزم های ارسال و دریافت پیام، امکان همکاری و برقراری ارتباط بین Actor ها فراهم شده و در نتیجه وظیفه‌ی اصلی و کلی نرم افزار با حرکت در یک خط لوله و عبور از Actor های مختلف به صورت موازی و همزمان انجام خواهد شد. با توجه به اینکه هر یک از پیام‌های وارده به یک Actor در یک thread جداگانه اجرا می‌شود، امکان اینکه در یک لحظه چندین Thread در یک Actor در حال اجرا باشند و جود دارد و درنتیجه باید مکانیزم‌هایی وجود داشته باشد که تضمین کند پیام های وارد شده به خط لوله، به ترتیب معین شده، اجرا و از خط لوله خارج می‌شوند.

در این مدل هر یک از Actor ها می توانند به صورت توزیع شده و بر روی سروری مجزا اجرا شوند. خوشبختانه فریمورک های متفاوت و بسیار قوی جهت توسعه به روش Actor Base وجود دارند؛ به عنوان مثال TPL DataFlow در .Net یکی از نمونه‌های ساده آن بوده که در سال 2012 توسط Microsoft معرفی شد و Akka هم یک نمونه‌ی بسیار پخته‌تر و در بستر جاوا مطرح می‌باشد که پیاده سازی دات نتی آن هم با نام Akka.net موجود است. Erlang نیز محصول Ericsson بوده و دنیای خاص خود را دارد.

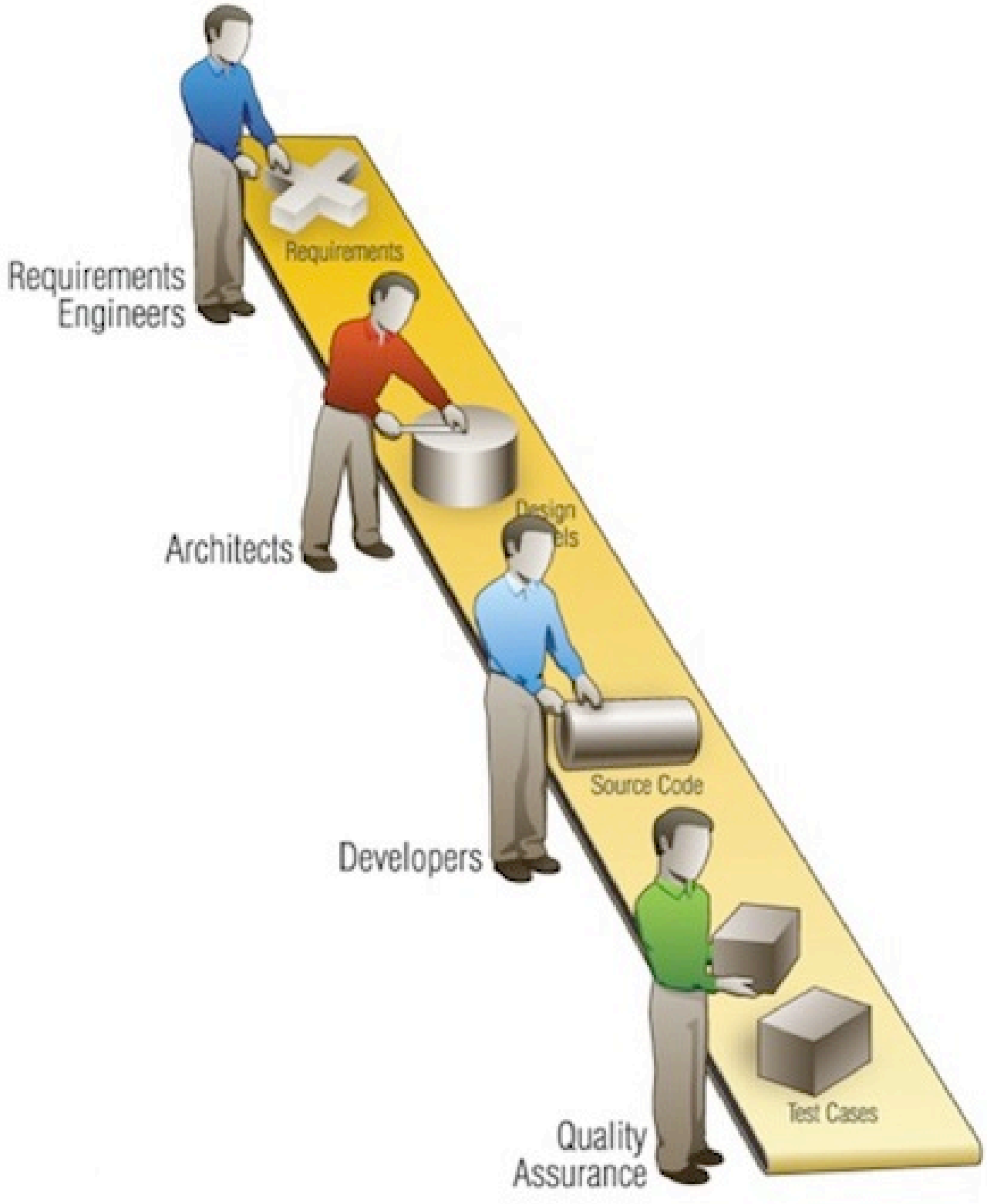
در این روش وظیفه توسعه دهنده این است که اولاً یک خط لوله از اکتورها را تشکیل داده (کانفیگ) و یک عمل بزرگ را به چندین عمل کوچک‌تر تقسیم نموده و هر کدام را به یک اکتور جهت اجرا ارسال نماید. تصویر نمونه زیر یک خط لوله متشکل از 4 اکتور را نشان می‌دهد که از طریق ارسال پیام با یکدیگر در ارتباط هستند تا با همکاری یکدیگر عملی را انجام دهند. این ساختار، Pipeline یا خط لوله نامیده می‌شود.



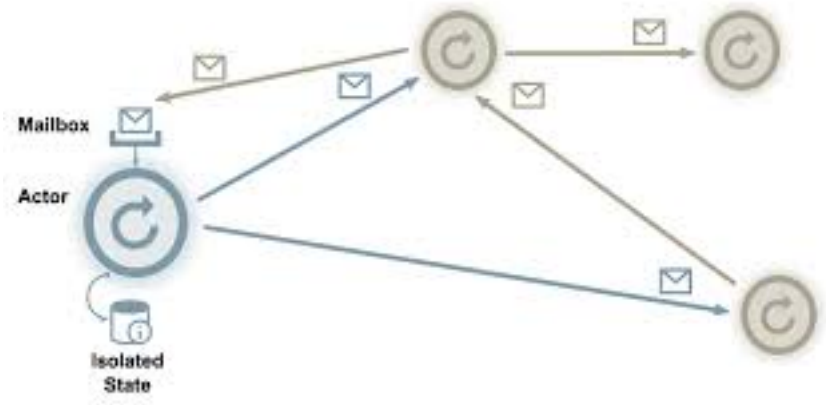
در قسمت بعدی با جزئیات بیشتر و با نمونه‌های عملی این روش را بررسی می‌کنیم.

عنوان: **معرفی Actor Based Programming و توسعه نرم افزار های مقیاس پذیر و دارای عملیات همزمان بسیار زیاد - قسمت دوم**
نویسنده: ایمان رحیمی نیا
تاریخ: ۱۳۹۴/۰۵/۲۲ ۱۴:۴۵
آدرس: www.dotnettips.info
گروه‌ها: Asynchronous Programming, Tpl DataFlow, Erlang, Akka, actor based programming, TPL

در قسمت قبل توضیحاتی راجع به مقدمات Actor Based Programming و کاربرد آن داده شد و چند framework نیز برای توسعه به این روش معرفی گردید. در این قسمت جزئیات بیشتری را از این روش توسعه، ارائه خواهیم داد.
خط تولید کارخانه‌ای را فرض کنید که در آن یک قطعه از ابتدای خط حرکت نموده و کارگران مستقر در خط تولید نیز هر کدام بنا به وظیفه‌ی خود، کاری را بر روی قطعه‌ی مورد نظر انجام می‌دهند؛ به طوریکه در انتهای خط تولید، آن قطعه‌ی اولیه، به یک محصول کامل تبدیل می‌شود.


Requirements
Requirements Engineers
Architects
Source Code
Developers
Test Cases
Quality Assurance

ایده‌ی Actor Based نیز هم از همین روش الهام گرفته است. با این تفاوت که بجای کارگران، Thread داریم و بجای قطعه نیز یک پیام یا object و بجای خط تولید نیز خط لوله یا pipeline را داریم. همانطور که در قسمت قبل اشاره کردم، وظیفه‌ی توسعه دهنده در این روش، طراحی یک خط لوله و نوشتن کد مربوط به هر thread است. به همین سادگی!
یعنی تمام پیچیدگی‌های مربوط به concurrency و مسائل فنی توسط یک framework مثل TPL DataFlow یا Akka کنترل و مدیریت می‌شود و توسعه دهنده با تمرکز بر روی مسئله‌ی خود، شروع به طراحی (کانفیگ) خط لوله و نوشتن کد مربوط به هر کدام از threadها می‌نماید.



تصویر بالا یک خط لوله را با چهار اکتور، نشان می‌دهد. می‌توان اینطور فرض نمود که هر اکتور یک mailbox دارد و اگر پیامی برای آن اکتور بفرستید، آن را پردازش نموده و کار مخصوص به خود را بر روی آن پیام انجام می‌دهد و سپس آن پیام را برای اکتور بعدی خود ارسال می‌کند. اکتور دوم نیز به همان ترتیب کار خود را انجام داده و پیام را به اکتور مابعد خود ارسال می‌کند و به این ترتیب، یک پیام در خط لوله حرکت نموده و فرآیند مربوطه انجام می‌شود. اگر دقت کنید یک فرق دیگر هم بین خط تولید کارخانه و این خط لوله وجود دارد و آن این است که این خط لوله به صورت گراف می‌باشد. یعنی اکتورها می‌توانند در ارتباط خود یک حلقه را تشکیل دهند و یا یک اکتور با چندین اکتور ارتباط مستقیم داشته باشد (مثل اکتور سمت چپ تصویر که با دو اکتور دیگر در ارتباط است).

خوب حالا که با مفاهیم خط لوله و اکتور آشنا شدیم، یک مسئله‌ی بسیار ساده را در نظر می‌گیریم و آن را با این روش حل می‌کنیم. فرض کنید یک رشته (string) داریم و می‌خواهیم عملیات زیر را بر روی آن به ترتیب انجام دهیم:

-1 فاصله‌های اضافی ابتدا و انتهای رشته حذف شود.

-2 اگر رشته یک کلمه‌ای است lowerCase شود.

-3 اگر رشته بیش از یک کلمه است، تمام کلمات، به جز کلمه‌ی اول، حذف شوند و سپس مرحله‌ی 2 بر روی آن انجام شود.

-4 نتیجه‌ی کار در خروجی نمایش داده شود.

حالا می‌خواهیم انجام هر یک از عملیات فوق را به یک اکتور سپرده و یک خط لوله را برای حل این مسئله طراحی کنیم. در قسمت بعدی به صورت عملی و با TPL DataFlow مایکروسافت این کار را انجام می‌دهیم.