KURVE
plataforma digital

Frameworks reactivos

Concepto

La programación reactiva es un paradigma enfocado en el trabajo con flujos de datos finitos o infinitos de manera asíncrona. Su concepción y evolución ha ido ligada a la publicación del Reactive Manifesto, que establecía las bases de los sistemas reactivos. La motivación detrás de este nuevo paradigma procede de la necesidad de responder a las limitaciones de escalado presentes en los modelos de desarrollo actuales, que se caracterizan por su desaprovechamiento del uso de la CPU debido al I/O, el sobreuso de memoria (enormes thread pools) y la ineficiencia de las interacciones bloqueantes.

El modelo de programación reactiva ha evolucionado de manera significativa desde su concepción en 2010. Las librerías que lo implementan se clasifican en generaciones de acuerdo con su grado de madurez:

- Generación 0: java.util.Observable/Observer proporcionaban la base de uso del patrón Observer del Gang of Four. Tienen los inconvenientes de su simplicidad y falta de opciones de composición.
- 1ª Generación: en 2010 Microsoft publica RX.NET, que en 2013 sería portado a entorno Java a través de la librería RxJava. Evolucionó el concepto de Observable/Observer incorporando la composición de operaciones, pero presentaba limitaciones arquitectónicas.
- 2ª Generación: se solucionan los problemas de backpressure y se introducen dos nuevas interfaces: Subscriber y Producer.
- 3ª y 4ª Generación: se caracterizan principalmente por haber eliminado la incompatibilidad entre las múltiples librerías del ecosistema reactive a través de la adopción de la especificación Reactive Streams, que fija dos clases base Publisher y Subscriber. Entran dentro de esta generación proyectos como RxJava 2.x, Project Reactor y Akka Streams.

Características principales

Los sistemas reactivos se caracterizan por ser:

- Responsivos: aseguran la calidad del servicio cumpliendo unos tiempos de respuesta establecidos.
- Resilientes: se mantienen responsivos incluso cuando se enfrentan a situaciones de error.
- Elásticos: se mantienen responsivos incluso ante aumentos en la carga de trabajo.
- Orientados a mensajes: minimizan el acoplamiento entre componentes al establecer interacciones basadas en el intercambio de mensajes de manera asíncrona.