BeSpoke.

REACTIVE ARCHITECTURE

Reactive Developer High Productivity Reactive Application Platform

The need to go Reactive

This architecture allows developers to build systems that are *event-driven, scalable, resilient and responsive*: delivering highly responsive user experiences with a real-time feel, backed by a scalable and resilient application stack, ready to be deployed on multicore and cloud computing architectures.

"Business environment today is so competitive that requires we are always ready to react to any events...reactive architecture give us a lot of edge to innovate as quickly as required"

Event Driven

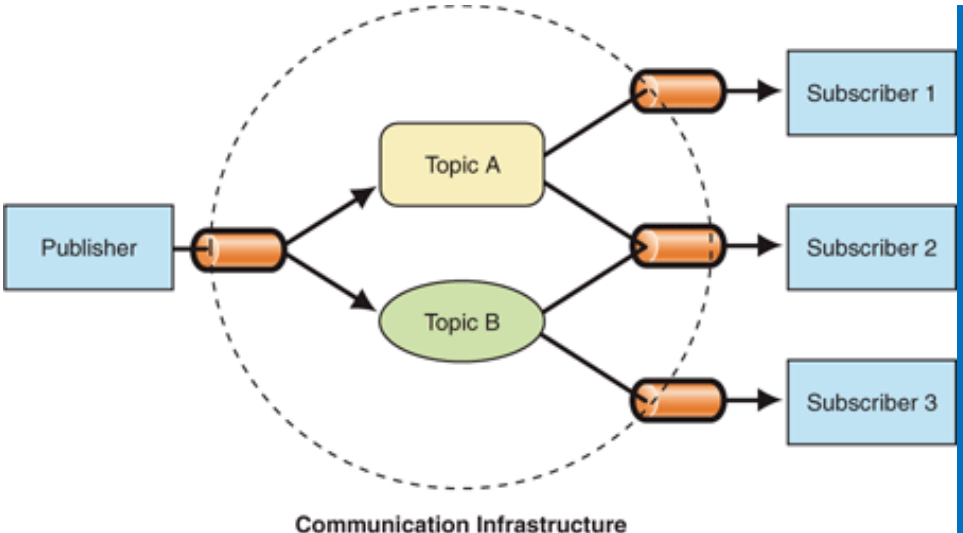
Every time your user do something, they produce events

Q: What is event?
A: a thing that happens, e.g. a button is clicked or an order is place, a patient is registered,

Q: What is Reactive?
A: Literally it's a state : Readily responsive to a stimulus (events).

Reactive Developer

Reactive Developer offers a set of tools and platform for enterprise to build highly scalable and resilient application in a loosely coupled architecture.



Topic subscription

More on Reactive

- Publisher and subscribers
- No code subscriber registration
- Innovate
- Make it scalable
- Resilient

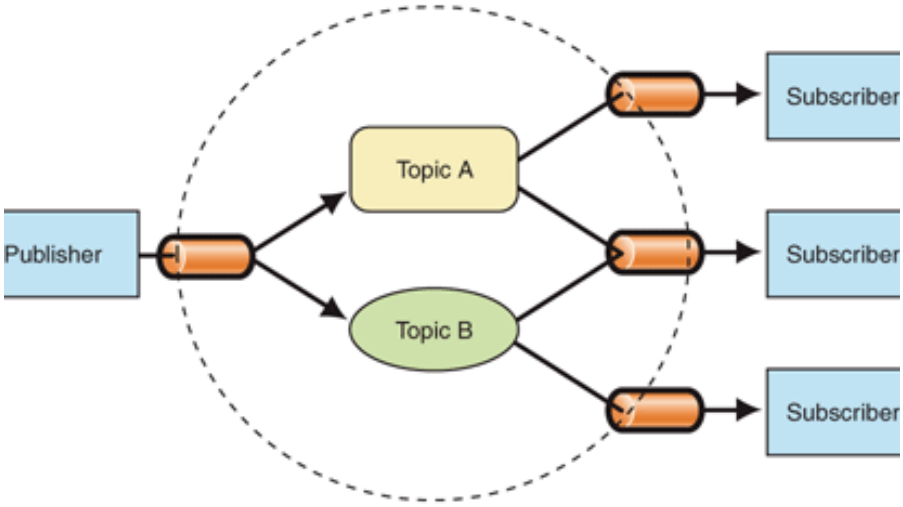


Photo Caption

“I’m afraid it will break something else”

This is the number one excuse given by developers when required to change to any dependencies in their code.

The reason, is that all their dependencies were hard coded together, running in the same event handler. Reactive Developer on the other hand wire the dependencies via loosely coupled publish and subscribe model. Any subscribers is an independent piece of functionality on it’s own.

Publisher and subscriber

Adopting Reactive Architecture *Publish and Subscribe* pattern allows your enterprise to organically grow your application in a more scalable manner. Applications developers need not to aware or worry of falling into the traps of related dependencies. They now could just focus on developing the part they are tasked to do.

Event subscriptions

All the related dependencies could be registered later in without any modification to existing application. Any changes to the dependencies could be deployed independently without the need to recompile the application.

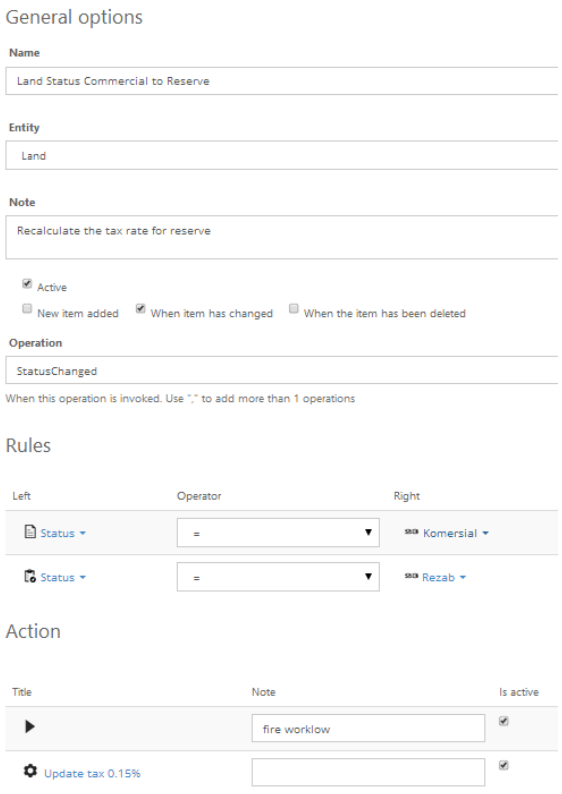
Traditional object oriented architecture

Developers are required to know about every dependencies up front during the design or development phase. This led to a very rigid architecture, hard to scale and high maintenance cost

```
public void PatientRegistered(Patient patient)
{
    database.InsertPatient(patient);
    database.UpdateRegistration(patient);

    finance.CreateAccount(patient);
    pharmacy.RegisterPrescription(patient);
    //
    // etc
}
```

Subscribe to event in Object Oriented



Subscribe to event in Reactive Developer

Event producer

It's where the event is originated, this could be your application like when a button is clicked, or your data model like when a patient is registered.

Topic and message

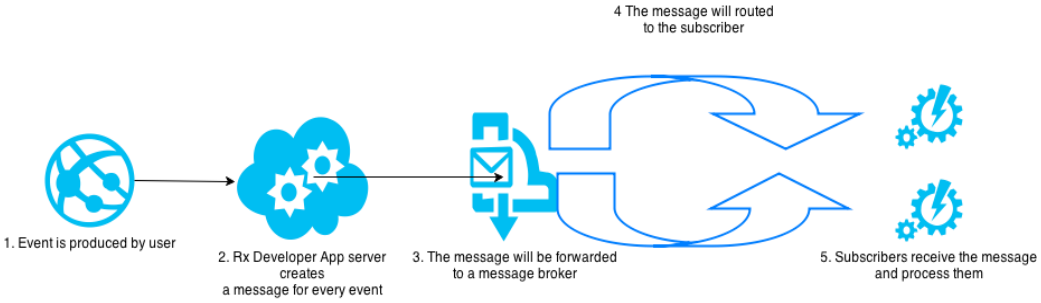
For every event produced a message containing the event information is created for example the patient data will be attached to the message body. The message will also be given a topic with tags e.g *button.clicked* or *patient.registered* or *patient.discharged*. Then this message will be sent to a message broker

Subscriber

Subscriber is a piece of event listener that register to a message broker and optionally specify any topics that they are interested in for example subscribing to *patient.registered*, it will receive every message about patient registered event. Subscribers also have the ability to subscribe to wildcard topic like *patient.** so that it will receive both events(patient registered and patient is discharged).

Message broker

It's a middle ware that is used to route the message



Message routing

How it works

Messaging is the heart of Reactive Developer communication. A message is created and sent to a message broker every time an event is produced. It can contains a topic, the message body and optionally message headers.

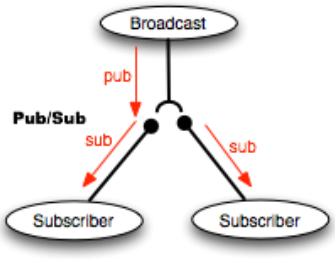
Messages sent to a broker, who will create a copy of the message for each subscribers and route them accordingly.

Once the message is delivered, the subscriber can process the message and acknowledge the broker that it has successfully processing the message.

If in case that the subscribers is unreachable, it's the brokers responsibility to keep the message until the subscribers is back online again.

Make it scalable

Since the application doesn't have to do much except creating and forwarding the message to a broker. It's much easier to scale



To make it scale, make it simple

Reactive Architecture is easy to scale because nothing is hard coded between networks of dependencies in a modern complex enterprise application we have today.

All the dependencies are configured as it's own unit of independent piece of software.

Reactive Developer makes it even easier for your enterprise to build Reactive Application, with it's no code , model driven approach and visual designers.

Contact Us

Give us a call for more information about our services and products

Bespoke Technology

(603) 77294424

sales@bespoke.com.my

Visit us on the web at
www.reactivedeveloper.com