MESSAGING AND RPC

Reactive Developer High Productivity Reactive Application Platform

Request and Response Model

HTTP is driven by stateless architecture with request and response model, what's normally known as Remote Procedure Call (RPC). Most of the time it's just what you need, when a user issue a request to your web application like click on a button, the browser will send a HTTP request to the server where it will ***process*** the request and return a response.

"A wise man once asked: What is the fastest way of processing a web request. His answer : Don't process it"

What's messaging Model

Messaging model relies on asynchronous messaging model, where a HTTP request is issued to the server, it *will not process* the request immediately, instead it issues a response ticket. Thus the client don't have to wait for the process to finish.

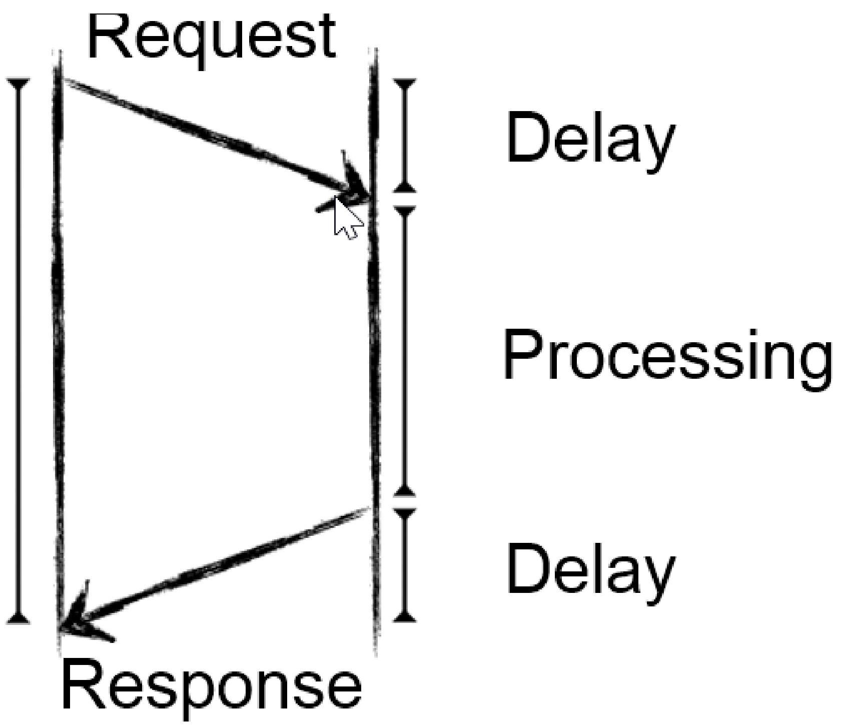
Reactive Developer

Reactive Developer offers a set of tools and platform for enterprise to build highly scalable and resilient application in based on messaging model.

Head of line blocking

More on Messaging

- Synchronous vs Asynchronous
- I/O Latency
- Asynchronous communication
- Make it scalable
- Resilient



RPC Synchronous Processing

Fast food kind of wait

If you ever queue up for a burger at a local fast food join. You'll have to wait for the person with his 3 kids in front of you to get all of his orders before they could take your request. Even you only want a burger.

It would be nice if they take the order at the start and call you when your orders are ready, so for a burger you don't have to queue behind a guy heading for weekend party.

Synchronous Processing

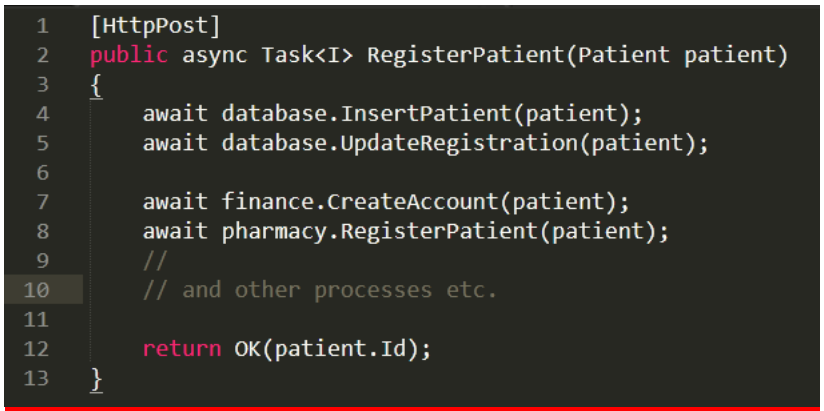
Most of modern enterprise application today are base on synchronous processing model, once a request is received it will be immediately processed, and it could happen on the main request thread sequentially or for some I/O intensive operation it will done on a separate I/O thread.

Head of line blocking

No matter how you do it, even on the separate I/O thread for I/O intensive operation like database call or network call, you make your users wait until all of these operations completed. This create a situation of wait and delay as the response cannot be generated before all the processes are finished.

Failures

Sometime the wait is not the only problems, if your request has to process many operations, a failure in one will cause the whole request to be aborted. This will not make a very happy user experience

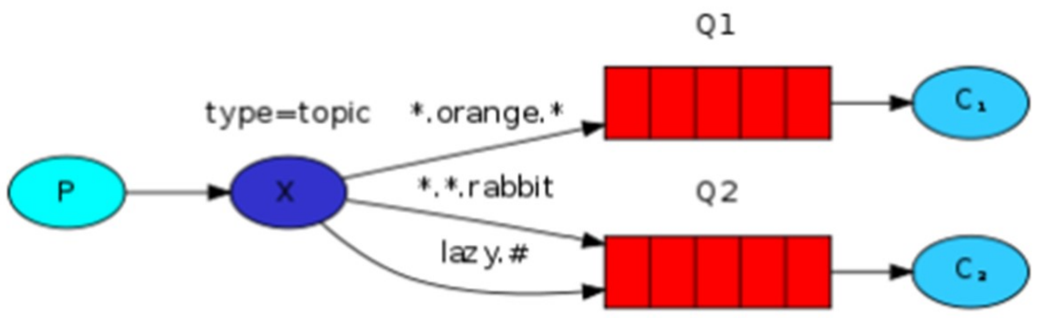


```
[HttpPost]
public async Task<I> RegisterPatient(Patient patient)
{
    await database.InsertPatient(patient);
    await database.UpdateRegistration(patient);

    await finance.CreateAccount(patient);
    await pharmacy.RegisterPatient(patient);
    //
    // and other processes etc.

    return OK(patient.Id);
}
```

Process a request in web application



A case of online retailers

Amazon.com is one of the biggest if not the biggest online retailers. They have to process as many as 500 orders per second. Now that's a lot of sales.

For every order fulfilled they have to process these operations
1. Payment verification according to the payment method.
2. Updates their inventory systems.
3. Updates their ERP for the fulfilled orders
4. Shipping information
5. and some more

If their customers have to wait for every operations to be completed then Amazon would be losing millions in sales.

With messaging, Amazon.com could fulfill their customers orders with relative ease.

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. A message can contains a topic, the message body and optionally message headers.

The message will be routed to the message broker who will take care of routing the message to it's subscribers. A copy of the message will be created for each subscribers.

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

Failures

Any failures can be contained, as this is not part of user's request pipeline they don't have to bear the frustration. The process can also be repeated or compensated accordingly.

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 messaging infrastructure, 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