Understanding Apache Camel

*“Apache Camel is an open source framework for message-oriented middleware with a rule-based routing and mediation engine that provides a Java object-based implementation of the Enterprise Integration Patterns using an application programming interface (or declarative Java domain-specific language) to configure routing and mediation rules. The domain-specific language means that Apache Camel can support type-safe smart completion of routing rules in an integrated development environment using regular Java code without large amounts of XML configuration files, though XML configuration inside Spring Framework is also supported…Camel is often used with Apache ServiceMix, Apache ActiveMQ and Apache CXF in service-oriented architecture projects….”*

Aaaaand breath, let’s break this down.  
How did we get here?

Remember the classic Design Patterns book, better known as the “Gang of Four (https://www.amazon.co.uk/Design-patterns-elements-reusable-object-oriented/dp/0201633612/ref=sr\_1\_3?ie=UTF8&qid=1508251144&sr=8-3&keywords=gang+of+four)”? This book introduced a set of simple and elegant solutions to solve common object orientated programming problems. Similarly, with the dawning of web services in the early 2000’s, another set of design patterns were realized. These patterns help facilitate integrations between n-tiered distributed applications by making use of messaging.   
  
To understand messaging, you’ll need to know the difference between synchronous and asynchronous communication. Think of a phone call as a means of synchronous communication, whereby the caller can only communicate if there is a receiver at the other end i.e. they’re not occupied. A voice mail however, can be described as asynchronous. When the receiver does not answer the phone, the caller can leave a message to be queued in their mailbox. The message will stay there for later consumption. The means of communication is a lot easier than trying to get the caller and receiver on the phone at the same time.