CQRS&&Axon

微服务是面向服务架构(SOA)架构风格的一种变体，将应用程序构建为一系列松散耦合的服务，在微服务体系结构中，服务是细粒度的，协议应该是轻量级的。将应用程序分解为不同的小型服务的好处是它提高了模块性，使应用程序易于理解、开发和测试。它使小型自主小组能够独立开发、部署和扩展各自的服务，从而实现并行发展，允许通过持续的重构来实现单个服务的体系结构，从而连续迭代和部署。但是随着服务数量的增加，服务的治理难度就会增加，这就带来了协议轻量级的概念。但是微服务开发事务性业务应用是一件具有挑战性的工作，CQRS(Command Query Responsibility Segregation)，命令查询职责分离的微服务开发方法可以实现可扩展且高效的查询。

# **CQRS概念**

命令查询职责分离模式（CQRS）从业务上分离命令（Command，增删改，对系统状态进行修改）和查询（Query，不会对系统状态进行修改）的行为，其基本思想，将任何一个对象的方法分为两类：

1. 命令，不返回任何结果，但会改变对象的状态
2. 查询，返回结果，但是不会改变对象的状态，对系统没有副作用

根据CQS(Command Query Separation)概念，任何方法都可以拆分为命令和查询，例如：

*private int i = 0;*

*private int Increase(int value){*

*i += value;*

*return i;*

*}*

上述方法，可以拆分为一个命令对变量i进行相加，同时又执行一个Query，即查询返回i的值，拆分如下：

*private int i = 0;*

*private void IncreaseCommand(int value){*

*i += value;*

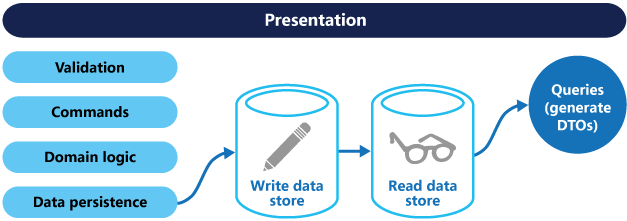
*}*

*private int queryValue(){*

*return i;*

*}*

将操作和查询分离可以更好的把握对象的细节，CQRS在查询和更新过程中使用不同的数据模型，从而使数据查询和修改操作分离，



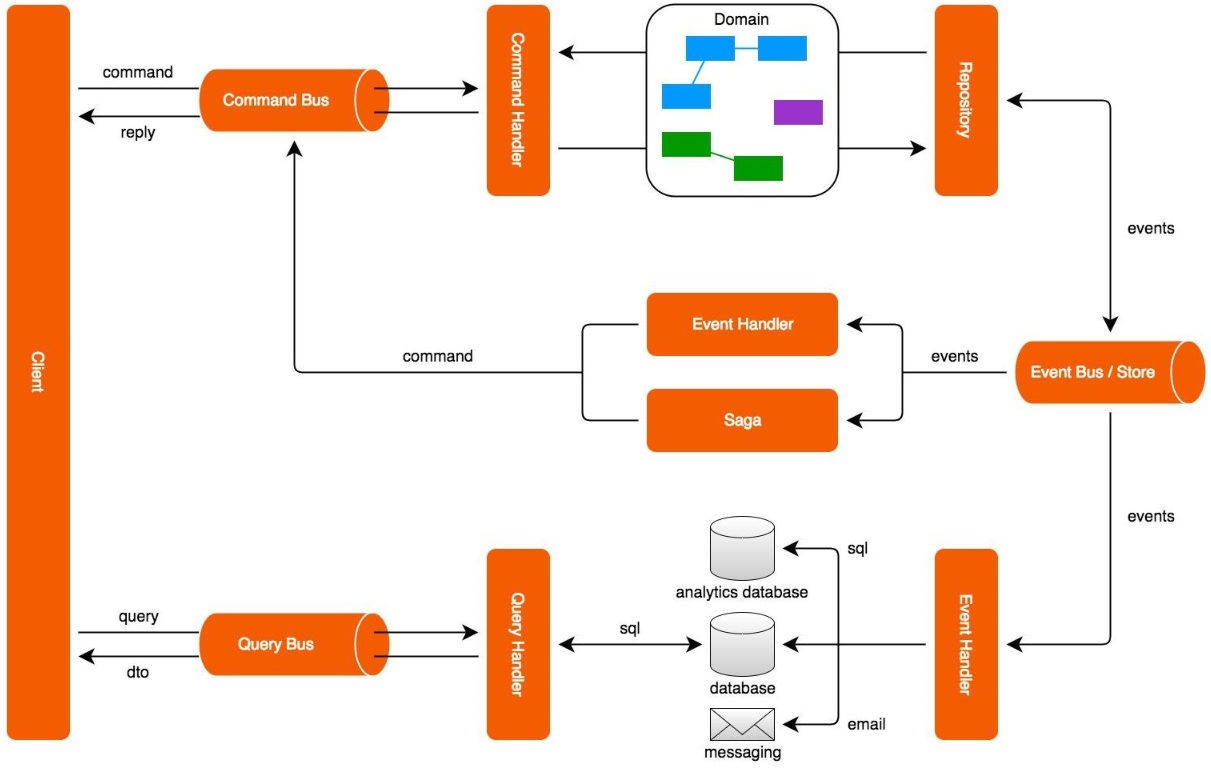
在业务中使用两个数据库，主数据库处理CUD操作，从库处理R，从库的结构和主库的结构可以完全一样，也可以不一样。使用CQRS模式，具有以下优点：

* 分工明确，可以负责不同的部分
* 将业务上的命令和查询的职责分离能够提高系统的性能、可扩展性和安全性，并且在系统演化中保持高度的灵活性，防止出现CRUD模式中对查询或者修改的某一方改动导致另一方查询问题
* 逻辑清晰，能够看到系统中哪些行为或者操作导致系统的状态变化
* 可以从数据驱动（Data-Driven）转到任务驱动或者事件驱动（Event-Driver），事件驱动或者任务驱动是实现CQRS模式的技术手段。

微服务是一种边界思维，从业务角度拆分（职责分离）业务，不同服务负责不同服务，每个微服务之间数据完全独立，它们之间的交互可以通过RPC、EDA(消息驱动)等方式。在微服务内部，可以使用CQRS/ES架构来实现。

# **Axon 框架**

Axon框架实现CQRS架构模式，提供聚合、存储以及事件总线（事件的调度机制）等构建模块来开发可伸缩、可扩展和可维护的应用程序，基于CQRS的事件驱动架构如下图：



UI组件向应用程序发送命令，并从应用程序中查询信息：

* Command Bus，命令总线：接收命令并路由到命令处理器（Command Handler），命令处理器响应特定类型的命令，然后根据命令的内容执行逻辑。
* Domian，领域模型：命令处理程序从存储库中检索域对象（聚合）并执行命令中的方法来更改其状态，聚合的状态变化导致产生领域事件。领域对象和聚合形成领域模型
* Repositories(仓储)：提供访问聚合。存储库通过聚合的唯一标识来查找聚合，在存储库中可存储聚合本身的状态，也可以在事件存储库中保存聚合经历的状态改变过程
* Query Bus，查询总线：接收查询并将它们路由到查询处理程序，查询处理程序注册在查询总线中，包括处理查询的类型以及查询响应的类型。查询和结果类型通常都是简单的只读DTO对象，其内容通常由用户界面的需求驱动

Axon提供了直接持久化聚合（使用对象关系映射，ORM）和事件溯源（event Sourcing）两种方式实现CQRS。其中在事件溯源中有以下概念：

* Event bus(事件总线)，分配事件到所有感兴趣的事件监听器Event Listener中，可以同步或者异步完成。异步事件调度允许命令执行返回和移交控制给用户，这些事件在后台被分派和处理，不必等待事件处理完成。
* Event Listener(事件监听器)，接收并处理事件，处理器将更新用于查询的数据源，而其他处理器则将消息发送到外部系统

Axon由多个针对CQRS特定问题域的模块组成，这些模块提供工具和组件来构建可伸缩的基础设施，每个模块通过jar的形式提供基本的API

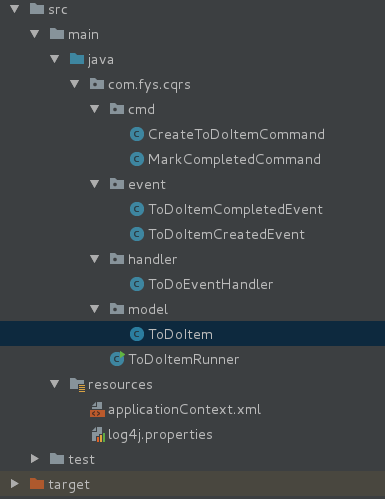
* 分布式CommandBus模块，可用于在多个节点上分发命令的实现，其带有连接这些节点的JGroups和Spring Cloud连接器
* Saga，事件监听器，管理业务事务，每个Saga实例负责管理一个业务事务
* AMQP模块，使用基于AMQP的消息代理为分发机制来构建EventBus
* Spring模块，在Spring应用程序上下文中配置Axon模块
* MongoDB模块，提供Event和Saga Store实现，将事件流和适配器存储在MongoDB数据库中
* Metrics模块，提供基于Codehale基本实现来收集监控信息

在使用Axon之前介绍其相关的概念

1. 消息，Axon的核心概念是通信，在组件间的通信都使用消息对象来完成。消息由唯一标识符，元数据和消息body组成，元数据描述发送的上下文，如安全信息。
2. 命令，描述改变应用程序状态的意图，实现为POJO，继承CommandMessage
3. 事件，描述应用程序中发送事件的对象，典型来源是聚合，在聚合中发生事件时引发另一个事件，事件可以是任何对象。在Axon将事件继承EventMessage，事件的实际类型取决于事件的来源，当一个事件由Aggregate引发时封装在DomainEventMessage中
4. 查询，描述对信息或状态的请求，在分配查询时客户端指示其想要的结果或者查询处理的程序
5. 工作单元，消息的处理被称为一个单元，其目的是协调处理消息（命令、事件或者查询）期间执行的操作。组件可以注册在工作单元每个阶段执行的操作，如onPrepareCommit或onCleanup。在Axom中，自动管理工作单元的生命周期

# **Axon的使用**

下面介绍Axon的使用示例，项目结构如下：



1. cmd，定义各种命令
2. event，定义各种事件
3. handler，定义事件处理器
4. model，相当于领域模型
5. 最外层的ToDoItemRunner相当于应用程序入口

使用Spring管理Beans，在applicationContext.xml中的定义如下：

*<?xml version="1.0" encoding="UTF-8"?>*

*<beans xmlns="http://www.springframework.org/schema/beans"*

*xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"*

*xmlns:axon="http://www.axonframework.org/schema/core"*

*xsi:schemaLocation="http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.axonframework.org/schema/core http://www.axonframework.org/schema/axon-core-2.0.xsd">*

*<axon:command-bus id="commandBus"/>*

*<axon:event-bus id="eventBus"/>*

*<axon:event-sourcing-repository*

*id="toDoRepository"*

*aggregate-type="com.fys.cqrs.model.ToDoItem"/>*

*<axon:aggregate-command-handler aggregate-type="com.fys.cqrs.model.ToDoItem"*

*id="toDoItemHandler"*

*repository="toDoRepository"*

*command-bus="commandBus"/>*

*<axon:filesystem-event-store base-dir="events" id="eventStore"/>*

*<axon:annotation-config/>*

*<bean class="com.fys.cqrs.handler.ToDoEventHandler"/>*

*<bean class="org.axonframework.commandhandling.gateway.CommandGatewayFactoryBean">*

*<property name="commandBus" ref="commandBus"/>*

*</bean>*

*</beans>*

测试程序如下：

*public class ToDoItemRunner {*

*private CommandGateway commandGateway;*

*public ToDoItemRunner(CommandGateway commandGateway) {*

*this.commandGateway = commandGateway;*

*}*

*public void run() {*

*final String itemId = UUID.randomUUID().toString();*

*commandGateway.send(new CreateToDoItemCommand(itemId, "Need to do this"));*

*}*

*public static void main(String[] args) {*

*ApplicationContext context =*

*new ClassPathXmlApplicationContext("applicationContext.xml");*

*ToDoItemRunner runner =*

*new ToDoItemRunner(context.getBean(CommandGateway.class));*

*runner.run();*

*}*

*}*

输出结果如下：

*We've got something to do :Need to do this ( ec3ceb3e-ecbe-4e45-a8cd-70d3a2e7d9ec )*

*We've complted a task:ec3ceb3e-ecbe-4e45-a8cd-70d3a2e7d9ec*

https://github.com/AxonFramework/Axon-trader

http://www.cnblogs.com/yjmyzz/archive/2016/03/03/cqrs-axonframework-sample.html

Axon博客，使用指南系列：

https://blog.csdn.net/wangli13860426642/article/details/80018222

http://www.axonframework.org/

CQRS概念：

http://www.infoq.com/cn/articles/microservices-aggregates-events-cqrs-part-1-richardson?utm\_source=infoq&utm\_campaign=user\_page&utm\_medium=link

http://www.infoq.com/cn/articles/microservices-aggregates-events-cqrs-part-2-richardson

http://www.cnblogs.com/yangecnu/p/Introduction-CQRS.html