RandomProducer

+ RandomProducer(producerId: String, topicType: String)
+ assignEvent(event: Event<T>, partitionList: List<Partition<T>>, partitionId: String): boolean

ManualProducer

+ ManualProducer(producerId: String, topicType: String)
+ assignEvent(event: Event<T>, partitionList: List<Partition<T>>, partitionId: String): boolean

Producer

- producerId: String
- topicType: String

+ Producer(producerId: String, topicType: String)
+ getProducerId(): String
+ getTopicType(): String
+ getType(): String
+ assignEvent(event: Event<T>, partitionList: List<Partition<T>>, partitionId: String): boolean

0..*

1

TributaryCLI

- tributary: TributaryService<T>
- commandType: String
- classType: String
- eventValueType: String

+ main(args: String[]): void
+ TributaryCLI()
+ handleInput(input: String): void
+ handleCreateInput(inputConfig: String[]): void
+ handleShowInput(inputConfig: String[]): void
+ handleDeleteInput(inputConfig: String[]): void
+ handleProduceInput(inputConfig: String[]): void
+ handleConsumeInput(inputConfig: String[]): void
+ parallelProduce(producers: List<Producer<T>>, topic: String, events: List<Event<T>>)
+ parallelConsume(consumers: List<Consumer<T>>, partitions: List<Partition<T>>)

TributaryCluster<T>

- producerList: List<Producer<T>>
- topicList: List<Topic<T>>
- consumerGroupList: List<ConsumerGroup<T>>
- validConsumerGroupBalancingMethods: List<String>
- mapper: ObjectMapper

+ TributaryCluster()
+ clearTributaryCluster(): void
+ showTopic(id: String): String
+ showConsumerGroup(id: String): String
+ showProducer(id: String): String
+ showAll(): void
+ createTopic(id: String, type: String): Topic<T>
+ createPartition(topicId: String, partitionId: String): Partition<T>
+ createConsumerGroup(consumerGroupId: String, topicId: String, balancingMethod: String): ConsumerGroup<T>
+ createConsumer(consumerGroupId: String, consumerId: String): Consumer<T>
+ createProducer(producerId: String, type: String, allocation: String): Producer<T>
+ deleteConsumer(consumerId: String): void
+ produceEvent(producerId: String, topicId: String, eventContentFilePath: String, partitionId: String, valueType: Class<T>): Event<T>
+ consumeEvent(consumerId: String, partitionId: String): Event<T>
+ consumeEvents(consumerId: String, partitionId: String, numberOfEvents: int): List<Event<T>>
+ setConsumerGroupRebalancing(consumerGroupId: String, balancingMethod: String): ConsumerGroup<T>
+ parseJsonToEvent(eventFilePath: String, classType: Class<T>): T
+ parallelProduce(producers: List<Producer<T>>, topic: String, events: List<Event<T>>)
+ parallelConsume(consumers: List<Consumer<T>>, partitions: List<Partition<T>>)
- findTopic(id: String): Topic<T>
- findPartition(id: String): Partition<T>
- findProducer(id: String): Producer<T>
- findConsumerGroup(id: String): ConsumerGroup<T>
- findConsumer(id: String): Consumer<T>
- setProducerList(producerList: List<Producer<T>>): void
- setTopicList(topicList: List<Topic<T>>): void
- setConsumerGroupList(consumerGroupList: List<ConsumerGroup<T>>): void

Topic<T>

- topicId: String
- topicType: String
- partitionList: List<Partition<T>>

+ Topic(topicId: String, topicType: String)
+ getTopicType(): String
+ getTopicId(): String
+ getType(): String
+ getPartitionList(): List<Partition<T>>

0..*

1

Partition<T>

- partitionId: String
- eventQueue: Queue<Event<T>>

+ Partition(partitionId: String)
+ getEventQueue(): Queue<Event<T>>
+ getPartitionId(): String

0..*

Event<T>

- eventId: String
- header: Header<T>
- key: String
- value: T

+ Event(eventId: String, header: Header<T>, key: String, value: T)
+ getHeader(): Header<T>
+ getEventId(): String
+ getDateTimeCreated(): long
+ getPayloadType(): String
+ getKey(): String
+ getValue(): T

Header<T>

- headerId: String
- dateTimeCreated: long
- payloadType: String

+ Header(headerId: String, valueType: Class<T>)
+ getHeaderId(): String
+ getDateTimeCreated(): long
+ getPayloadType(): String
+ setPayloadType(payloadType: String): void

ConsumerGroup<T>

- consumerGroupId: String
- topicId: String
- balancingMethod: String
- consumerList: List<Consumer<T>>

+ ConsumerGroup(consumerGroupId: String, topicId: String, balancingMethod: String)
+ addConsumer(c: Consumer<T>): void
+ getConsumerList(): List<Consumer<T>>
+ getConsumerGroupId(): String
+ getTopicId(): String
+ getBalancingMethod(): String
+ setBalancingMethod(balancingMethod: String): void

0..*

1

Consumer<T>

- consumerId: String
- consumerGroupId: String
- consumedEventList: List<Event<T>>

+ Consumer(consumerGroupId: String, consumerId: String)
+ consume(p: Partition<T>): Event<T>
+ getConsumerGroupId(): String
+ getConsumerId(): String
+ getConsumedEventList(): List<Event<T>>