package com.twitter.recosinjector.event\_processors

import com.twitter.eventbus.client.{EventBusSubscriber, EventBusSubscriberBuilder}

import com.twitter.finagle.mtls.authentication.ServiceIdentifier

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.logging.Logger

import com.twitter.scrooge.{ThriftStruct, ThriftStructCodec}

import com.twitter.util.Future

/\*\*

\* Main processor class that handles incoming EventBus events, which take forms of a ThriftStruct.

\* This class is responsible for setting up the EventBus streams, and provides a processEvent()

\* where child classes can decide what to do with incoming events

\*/

trait EventBusProcessor[Event <: ThriftStruct] {

private val log = Logger()

implicit def statsReceiver: StatsReceiver

/\*\*

\* Full name of the EventBus stream this processor listens to

\*/

val eventBusStreamName: String

/\*\*

\* the thriftStruct definition of the objects passed in from the EventBus streams, such as

\* TweetEvent, WriteEvent, etc.

\*/

val thriftStruct: ThriftStructCodec[Event]

val serviceIdentifier: ServiceIdentifier

def processEvent(event: Event): Future[Unit]

private def getEventBusSubscriberBuilder: EventBusSubscriberBuilder[Event] =

EventBusSubscriberBuilder()

.subscriberId(eventBusStreamName)

.serviceIdentifier(serviceIdentifier)

.thriftStruct(thriftStruct)

.numThreads(8)

.fromAllZones(true) // Receives traffic from all data centers

.skipToLatest(false) // Ensures we don't miss out on events during restart

.statsReceiver(statsReceiver)

// lazy val ensures the subscriber is only initialized when start() is called

private lazy val eventBusSubscriber = getEventBusSubscriberBuilder.build(processEvent)

def start(): EventBusSubscriber[Event] = eventBusSubscriber

def stop(): Unit = {

eventBusSubscriber

.close()

.onSuccess { \_ =>

log.info(s"EventBus processor ${this.getClass.getSimpleName} is stopped")

}

.onFailure { ex: Throwable =>

log.error(ex, s"Exception while stopping EventBus processor ${this.getClass.getSimpleName}")

}

}

}