package com.twitter.product\_mixer.component\_library.module

import com.google.inject.Provides

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.Memcached

import com.twitter.finagle.Resolver

import com.twitter.finagle.memcached.protocol.Command

import com.twitter.finagle.memcached.protocol.Response

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

import com.twitter.finagle.mtls.client.MtlsStackClient.\_

import com.twitter.finagle.param.HighResTimer

import com.twitter.finagle.service.RetryExceptionsFilter

import com.twitter.finagle.service.RetryPolicy

import com.twitter.finagle.service.StatsFilter

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.inject.TwitterModule

import com.twitter.storehaus.ReadableStore

import com.twitter.timelines.impressionstore.store.TweetImpressionsStore

import com.twitter.timelines.impressionstore.thriftscala.ImpressionList

import javax.inject.Singleton

object TweetImpressionStoreModule extends TwitterModule {

private val TweetImpressionMemcacheWilyPath = "/s/cache/timelines\_impressionstore:twemcaches"

private val tweetImpressionLabel = "timelinesTweetImpressions"

@Provides

@Singleton

def provideTimelineTweetImpressionStore(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): ReadableStore[Long, ImpressionList] = {

val scopedStatsReceiver = statsReceiver.scope("timelinesTweetImpressions")

// the below values for configuring the Memcached client

// are set to be the same as Home timeline's read path to the impression store.

val acquisitionTimeoutMillis = 200.milliseconds

val requestTimeoutMillis = 300.milliseconds

val numTries = 2

val statsFilter = new StatsFilter[Command, Response](scopedStatsReceiver)

val retryFilter = new RetryExceptionsFilter[Command, Response](

retryPolicy = RetryPolicy.tries(

numTries,

RetryPolicy.TimeoutAndWriteExceptionsOnly

.orElse(RetryPolicy.ChannelClosedExceptionsOnly)

),

timer = HighResTimer.Default,

statsReceiver = scopedStatsReceiver

)

val client = Memcached.client

.withMutualTls(serviceIdentifier)

.withSession

.acquisitionTimeout(acquisitionTimeoutMillis)

.withRequestTimeout(requestTimeoutMillis)

.withStatsReceiver(scopedStatsReceiver)

.filtered(statsFilter.andThen(retryFilter))

.newRichClient(

dest = Resolver.eval(TweetImpressionMemcacheWilyPath),

label = tweetImpressionLabel

)

TweetImpressionsStore.tweetImpressionsStore(client)

}

}