package com.twitter.home\_mixer.module

import com.google.inject.name.Named

import com.google.inject.Provides

import com.twitter.conversions.DurationOps.RichDuration

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.home\_mixer.param.HomeMixerInjectionNames.TweetypieStaticEntitiesCache

import com.twitter.inject.TwitterModule

import com.twitter.product\_mixer.shared\_library.memcached\_client.MemcachedClientBuilder

import com.twitter.servo.cache.FinagleMemcache

import com.twitter.servo.cache.KeyTransformer

import com.twitter.servo.cache.KeyValueTransformingTtlCache

import com.twitter.servo.cache.ObservableTtlCache

import com.twitter.servo.cache.Serializer

import com.twitter.servo.cache.ThriftSerializer

import com.twitter.servo.cache.TtlCache

import com.twitter.tweetypie.{thriftscala => tp}

import javax.inject.Singleton

import org.apache.thrift.protocol.TCompactProtocol

object TweetypieStaticEntitiesCacheClientModule extends TwitterModule {

private val ScopeName = "TweetypieStaticEntitiesMemcache"

private val ProdDest = "/srv#/prod/local/cache/timelinescorer\_tweet\_core\_data:twemcaches"

private val tweetsSerializer: Serializer[tp.Tweet] = {

new ThriftSerializer[tp.Tweet](tp.Tweet, new TCompactProtocol.Factory())

}

private val keyTransformer: KeyTransformer[Long] = { tweetId => tweetId.toString }

@Provides

@Singleton

@Named(TweetypieStaticEntitiesCache)

def providesTweetypieStaticEntitiesCache(

statsReceiver: StatsReceiver,

serviceIdentifier: ServiceIdentifier

): TtlCache[Long, tp.Tweet] = {

val memCacheClient = MemcachedClientBuilder.buildMemcachedClient(

destName = ProdDest,

numTries = 1,

numConnections = 1,

requestTimeout = 50.milliseconds,

globalTimeout = 100.milliseconds,

connectTimeout = 100.milliseconds,

acquisitionTimeout = 100.milliseconds,

serviceIdentifier = serviceIdentifier,

statsReceiver = statsReceiver

)

mkCache(new FinagleMemcache(memCacheClient), statsReceiver)

}

private def mkCache(

finagleMemcache: FinagleMemcache,

statsReceiver: StatsReceiver

): TtlCache[Long, tp.Tweet] = {

val baseCache: KeyValueTransformingTtlCache[Long, String, tp.Tweet, Array[Byte]] =

new KeyValueTransformingTtlCache(

underlyingCache = finagleMemcache,

transformer = tweetsSerializer,

underlyingKey = keyTransformer

)

ObservableTtlCache(

underlyingCache = baseCache,

statsReceiver = statsReceiver.scope(ScopeName),

windowSize = 1000,

name = ScopeName

)

}

}