package com.twitter.home\_mixer.module

import com.google.inject.Provides

import com.twitter.conversions.DurationOps.\_

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

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.home\_mixer.{thriftscala => t}

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.Serializer

import com.twitter.servo.cache.ThriftSerializer

import com.twitter.servo.cache.TtlCache

import com.twitter.timelines.model.UserId

import org.apache.thrift.protocol.TCompactProtocol

import javax.inject.Singleton

object ScoredTweetsMemcacheModule extends TwitterModule {

private val ScopeName = "ScoredTweetsCache"

private val ProdDestName = "/srv#/prod/local/cache/home\_scored\_tweets:twemcaches"

private val StagingDestName = "/srv#/test/local/cache/twemcache\_home\_scored\_tweets:twemcaches"

private val scoredTweetsSerializer: Serializer[t.ScoredTweetsResponse] =

new ThriftSerializer[t.ScoredTweetsResponse](

t.ScoredTweetsResponse,

new TCompactProtocol.Factory())

private val userIdKeyTransformer: KeyTransformer[UserId] = (userId: UserId) => userId.toString

@Singleton

@Provides

def providesScoredTweetsCache(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): TtlCache[UserId, t.ScoredTweetsResponse] = {

val destName = serviceIdentifier.environment.toLowerCase match {

case "prod" => ProdDestName

case \_ => StagingDestName

}

val client = MemcachedClientBuilder.buildMemcachedClient(

destName = destName,

numTries = 2,

numConnections = 1,

requestTimeout = 200.milliseconds,

globalTimeout = 400.milliseconds,

connectTimeout = 100.milliseconds,

acquisitionTimeout = 100.milliseconds,

serviceIdentifier = serviceIdentifier,

statsReceiver = statsReceiver.scope(ScopeName)

)

val underlyingCache = new FinagleMemcache(client)

new KeyValueTransformingTtlCache(

underlyingCache = underlyingCache,

transformer = scoredTweetsSerializer,

underlyingKey = userIdKeyTransformer

)

}

}