package com.twitter.home\_mixer.module

import com.google.inject.Provides

import com.twitter.conversions.DurationOps.\_

import com.twitter.finagle.Memcached

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

import com.twitter.finagle.stats.StatsReceiver

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

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

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

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

import com.twitter.inject.TwitterModule

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

import com.twitter.servo.cache.FinagleMemcacheFactory

import com.twitter.servo.cache.Memcache

import javax.inject.Named

import javax.inject.Singleton

object MemcachedFeatureRepositoryModule extends TwitterModule {

// This must match the respective parameter on the write path. Note that servo sets a different

// hasher by default. See [[com.twitter.hashing.KeyHasher]] for the list of other available

// hashers.

private val memcacheKeyHasher = "ketama"

@Provides

@Singleton

@Named(TimelinesRealTimeAggregateClient)

def providesTimelinesRealTimeAggregateClient(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Memcache = {

val rawClient = MemcachedClientBuilder.buildRawMemcachedClient(

numTries = 3,

numConnections = 1,

requestTimeout = 100.milliseconds,

globalTimeout = 300.milliseconds,

connectTimeout = 200.milliseconds,

acquisitionTimeout = 200.milliseconds,

serviceIdentifier = serviceIdentifier,

statsReceiver = statsReceiver

)

buildMemcacheClient(rawClient, "/s/cache/timelines\_real\_time\_aggregates:twemcaches")

}

@Provides

@Singleton

@Named(HomeAuthorFeaturesCacheClient)

def providesHomeAuthorFeaturesCacheClient(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Memcache = {

val cacheClient = MemcachedClientBuilder.buildRawMemcachedClient(

numTries = 2,

numConnections = 1,

requestTimeout = 150.milliseconds,

globalTimeout = 300.milliseconds,

connectTimeout = 200.milliseconds,

acquisitionTimeout = 200.milliseconds,

serviceIdentifier = serviceIdentifier,

statsReceiver = statsReceiver

)

buildMemcacheClient(cacheClient, "/s/cache/timelines\_author\_features:twemcaches")

}

@Provides

@Singleton

@Named(TwhinAuthorFollowFeatureCacheClient)

def providesTwhinAuthorFollowFeatureCacheClient(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Memcache = {

val cacheClient = MemcachedClientBuilder.buildRawMemcachedClient(

numTries = 2,

numConnections = 1,

requestTimeout = 150.milliseconds,

globalTimeout = 300.milliseconds,

connectTimeout = 200.milliseconds,

acquisitionTimeout = 200.milliseconds,

serviceIdentifier = serviceIdentifier,

statsReceiver = statsReceiver

)

buildMemcacheClient(cacheClient, "/s/cache/home\_twhin\_author\_features:twemcaches")

}

@Provides

@Singleton

@Named(RealTimeInteractionGraphUserVertexClient)

def providesRealTimeInteractionGraphUserVertexClient(

serviceIdentifier: ServiceIdentifier,

statsReceiver: StatsReceiver

): Memcache = {

val cacheClient = MemcachedClientBuilder.buildRawMemcachedClient(

numTries = 2,

numConnections = 1,

requestTimeout = 150.milliseconds,

globalTimeout = 300.milliseconds,

connectTimeout = 200.milliseconds,

acquisitionTimeout = 200.milliseconds,

serviceIdentifier = serviceIdentifier,

statsReceiver = statsReceiver

)

buildMemcacheClient(cacheClient, "/s/cache/realtime\_interactive\_graph\_prod\_v2:twemcaches")

}

private def buildMemcacheClient(cacheClient: Memcached.Client, dest: String): Memcache =

FinagleMemcacheFactory(

client = cacheClient,

dest = dest,

hashName = memcacheKeyHasher

)()

}